Journal of Artificial Intelligence Research 59 (2017) 1-58 Submitted 10/16; published 05/17

Indirect Causes in Dynamic Bayesian Networks Revisited

Alexander Motzek MOTZEK@IFIS.UNI-LUEBECK.DE
Ralf Möller MOELLER@UNI-LUEBECK.DE
Universität zu Lübeck
Institut für Informationssysteme
23562 Lübeck, Germany

Abstract

Modeling causal dependencies often demands cycles at a coarse-grained temporal scale. If Bayesian networks are to be used for modeling uncertainties, cycles are eliminated with dynamic Bayesian networks, spreading indirect dependencies over time and enforcing an infinitesimal resolution of time. Without a "causal design," i.e., without anticipating indirect influences appropriately in time, we argue that such networks return spurious results. By identifying activator random variables, we propose activator dynamic Bayesian networks (ADBNs) which are able to rapidly adapt to contexts under a causal use of time, anticipating indirect influences on a solid mathematical basis using familiar Bayesian network semantics. ADBNs are well-defined dynamic probabilistic graphical models allowing one to model cyclic dependencies from local and causal perspectives while preserving a classical, familiar calculus and classically known algorithms, without introducing any overhead in modeling or inference.

1. Introduction

Dynamic Bayesian networks (DBNs) are an extension to Bayesian networks motivated from two perspectives, on the one hand as a manifestation of cyclic dependencies after a long period of simulation-time, closely related to Markov models (Murphy, 2002), on the other hand as a stationary process repeated in fixed timeslices (Glesner & Koller, 1995) over a wall-clock time to reason about historical and future evolutions of processes. Considering Pearl and Russell (2003) who emphasize that Bayesian networks should be a direct representation of the world instead of a reasoning process, Murphy's and Glesner's views seem to be conflicting: a stationary model repeated over a wall-clock time with cyclic dependencies would expand to infinity already for one timeslice. Therefore, e.g., Jaeger (2001) or Glesner and Koller use a strict order of dependencies s.t. state variables of time t are only dependent of states at $t-1$. Unfortunately, this means that evidence at a certain time point does not affect states at this time point, but one slice later. We argue and show that this limits the causal expressiveness of Bayesian networks.

In the extreme form of a directed dynamic probabilistic graphical model (DPGM), each random variable is locally and causally seen as dependent on every other random variable of one timestep. When using DBNs to allow for local specifications and intuitive interpretation of parameters, i.e., conditional probability distributions, there is no option to causally correctly leave all dependencies in the same timestep as dependencies would cause cycles. Therefore, random variables can only be dependent on variables from a previous timestep. However, we show in this article that models with such bent dependencies are severely at odds with

causality, as (i) the temporal causality is, simply, represented inaccurately, (ii) no indirect effects are considered at a particular timestep, enforcing an infinitesimal resolution of time adjusted to a reasoning process, instead of a time modeled for a world representation, and (iii) return spurious results under a coarser time granularity. These restrictions severely limit the usage and expressivity of DBNs.

To circumvent these limitations, basically three options are available. As investigated by Boutilier, Friedman, Goldszmidt, and Koller (1996), variables might be independent in certain contexts, which would allow for a causally correct network generation from rules such as those presented by Glesner and Koller (1995) or by Ngo and Haddawy (1997). However, rules will often need to be designed with a procedural view, degrading a BN to a procedural tool in a reasoning process, rather than designing it as a first-class declarative representation. Further, such rules would inherently be cyclic and might cause additional problems as stated by Ngo, Haddawy, and Helwig (1995). A second option would be to heavily restrict a DBN to specialized observation sets, e.g., to "single observations at a time" as done by Sanghai, Domingos, and Weld (2005), s.t. no indirect causes need to be considered. Obtaining only single observations during one timeslice again implies that observations are made at an infinitesimal resolution of time. Finally, a BN could be designed from a reasoning perspective, where edges solely identify correlation instead of causation, which, however, leads to an inherently non-local model. Such non-local models become impractical as a first-class declarative representation, as parameters neither bear any meaning nor provide any intuitive interpretation for a human.

We overcome the above-mentioned discrepancies and limitations by reconsidering fundamental aspects of dynamic Bayesian networks by which we increase the expressiveness of DBNs while remaining practically useable and locally understandable. The contributions of this article can, therefore, be summarized as follows. By considering DBNs in which some random variables show to have an activator nature, we prove that such DBNs, called ADBNs, are subject to a different acyclicity constraint, which allows an ADBN to rapidly adapt to contexts, by which such DBNs anticipate all indirect influences on a solid mathematical basis using familiar Bayesian network semantics, without requiring an infinitesimal resolution of time. Such ADBNs can be based on cyclic graphs using completely classical random variables, classical CPDs. This is achieved without introducing any novel calculus, and without introducing any overhead in exact or approximate inference. This broadens the expressiveness of DBN modeling so that applications are supported in which causal models naturally arise and cyclic dependencies emerge through local views on (in)dependencies. Example applications are automatic learning of causal influences from coarse observation sets and—as a long-term goal—finding causally correct explanations and relations in knowledge bases requiring context-aware interpretations and anticipations of causal chains, e.g., DeepQA (Ferrucci et al., 2010) or the Knowledge Vault (Dong et al., 2014).

1.1 Remainder of this Article

In Section 2 we informally outline an occurring problem when modeling domains from local perspectives as envisioned in Bayesian network with directed and causally created dependency relations. We show how Bayesian networks are used for modeling causal relationships and argue that cyclic dependencies arise naturally in this context. We discuss

preliminaries on DBNs and context-specific independencies as introduced by Boutilier et al. (1996) and Haddawy, Helwig, Ngo, and Krieger (1995) in Section 3 and identify an eminent problem of causality in DBNs that demands cyclic dependencies in every timestep of a DBN. By considering DBNs in which some random variables show to have an activator nature, we introduce Activator Dynamic Bayesian Networks (ADBNs) in Section 4. We show that such DBNs remain well-defined for cyclic dependencies—work that has been partially published at IJCAI 2015 by Motzek and Möller (2015b) for which theory has significantly advanced in Sections 5.4, 6 and 7 and that has been substantiated with further examples, an extended discussion of related work, as well as more detailed explanations and derivations. We investigate common queries and associated problems in ADBNs such as filtering and smoothing in Section 5, show that cyclic dependencies are required for anticipating indirect influences, and show that cycles in ADBNs do not introduce any new calculus or overhead for solving commonly known problems in DBNs. We substantiate our claims by providing experimental results for exact inference. Section 6 is dedicated to an introduction and derivation of an approximate inference techniques for (A)DBNs, which shows that approximate filtering remains possible and remains possible under a bounded error over time. In Section 7 we revisit the semantics of Bayesian networks and discuss models without commitments to structures, where effectively multiple joint distributions by one (D)PGM are represented. We discuss our results and relations to previous work in Section 8. Section 9 comes to a conclusion and gives an outlook for future work.

2. Cyclic Dependencies and Causality in Bayesian Networks

Pearl and Russell (2003) emphasize that Bayesian networks should be a direct representation of the world instead of a reasoning process. Essentially, a Bayesian network is "only" a more compact representation of a joint probability distribution of random variables. However, we strongly agree with Pearl and Russell and see a Bayesian network as much more powerful model, and not just as a reasoning tool for efficiently solving some probabilistic inference problems; We believe in Bayesian networks as a representation of a domain as it is directly evident in the real world, where all entities of the domain are represented in the Bayesian network, such that all questions regarding the domain are formulated as inference problems in the Bayesian network, which, in turn, immediately provides the answer by its semantics. To allow for this, directions of edges in a probabilistic graphical model must represent causality and identify cause→effect relationships, and should not be adjusted to restrictions opposed by a reasoning framework to simply solve some inference problem. While edge directions in probabilistic graphical models are irrelevant for the expressivity, i.e., ability to represent full joint probability distributions, only a *causal* edge direction allows for a desired intuitive, directly understandable, and local parametrization of conditional probability distributions (CPDs) in Bayesian networks. Pearl and Russell introduce local implications of parameters, i.e., CPDs, in Bayesian networks as their local semantics, to which we accredit the local understandability and direct, intuitive interpretability of CPDs as well. The product of those local parameters defines the global semantics of a Bayesian network without the need of any global normalization factors which allows one to locally and individually interpret these values. We consider these local semantics of CPDs as a highly valuable property of probabilistic graphical models, as they exactly allow one to directly represent knowledge directly obtained from multiple experts from different expertises.

As mentioned before, Bayesian networks provide desired local semantics, and every random variable is designed by only considering its parents, i.e., its direct causes, in the form of a CPD and the local CPD has a unique (local) semantics and neither requires a consideration of (conditionally) independent random variables nor implications of other locally defined CPDs. However, Bayesian networks are classically only well-defined for directed acyclic graphs, but aforementioned local and causal views on dependencies often lead to *cyclic* dependencies as the following example shows.

Example 1 (Human interaction and emotions)**.** *Cause and effects of human interactions are often not uniquely identifiable without a further context and play a major role in Winograd challenges (see, e.g., Levesque, Davis, & Morgenstern, 2012). Seeing a crying person and hearing a joke being told can have two explanations: (a) a person is crying from laughter, because the joke was good or (b) a person was told a joke to cheer her up, because she was in a sad mood, crying and needs comforting. Modeling these relationships with a probabilistic graphical model, as shown in Figure 1, requires two random variables Joke and Crying to have a cyclic dependency. An actual causality, i.e., a direction of the edge, is only known in a further context, e.g., a funeral or a party being observed (Place). The fact that influence directions depend on contexts are, in fact, a key feature of Winograd challenges (cf. Levesque et al., 2012).*

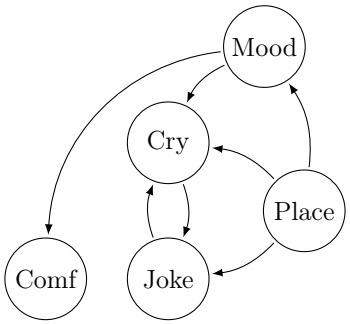

Figure 1: Modeling causal dependencies often requires cyclic dependencies. If one is supposed to reason over *Cry*ing, potential influences *of Joke* or *on Joke* must be considered. Without a link between *Joke* and *Cry*, the only possible explanation for crying at a party is that the person is *Mood* = *sad* or a person always cries at parties—usually a wrong assumption. And without a link from Cry to Joke, the only explanation for a Joke being told at a funeral is the funeral itself—usually a quite macabre assumption.

Example 1 represents an example for a cyclic dependency created due to local views on dependencies, and one must conclude that dependencies cannot be modeled causally correctly with acyclic PGMs. Nevertheless, a well-defined probabilistic graphical model for this domain domain would allow an artificial intelligence to precisely reason about multiple situations and decisions, e.g., is it appropriate to tell a joke?, why are people crying?, what is the mood of present people?, or, what scene is observed? Unfortunately, no straightforward solution in a Bayesian network formalism exists: Removing both edges of the alleged cyclic dependencies between *Cry* and *Joke* destroys causality, as the essential causes and effect relationships are removed: Jokes are not told at a funeral, just because of the funeral taking place; jokes

are told at a funeral to possibly cheer up crying, sad persons. Further, someone (usually) does not cry at a comedy event because of being at a comedy event; a person is crying at a comedy event because jokes (which are usually told at comedy events) were good and made a person cry. One could mimic an intended joint probability distribution by a single edge between *Cry* and *Joke*, e.g., only $Cry \rightarrow Joke$, which, however, destroys the local semantics: The local CPD of *Joke* must then be designed w.r.t. an intended influence of $Joke \rightarrow Cry$, i.e., a CPD is not solely designed with respect to causal cause→effect relationships from direct parents, but also with an inverse "reasoning" view on effect→cause relationships. Note that, such allegedly cyclic dependencies between two variables need not be evident directly, but can emerge from a constellation of many random variables in larger models for complex domains.

By moving away from directed PGMs, one is able to represent the outlined domain as a completely undirected PGM or as a partially directed PGM. For example, by sacrificing the identification of causality, the alleged cyclic edges between *Joke* and *Cry* could be represented by one undirected edge associated with a factor for uncertainty. However, such factors, sometimes called "potentials," bear no local meaning and are not interpretable. Moreover, local semantics are lost, as first a global normalization factor on the complete model enables a certain degree of interpretation.

By sacrificing a Bayesian network as a first-class representation, the joke-cry-domain could be represented by some reasoning frameworks designed via rules, e.g., Problog by de Raedt et al. (2007) or earlier work by Glesner and Koller (1995) or by Ngo and Haddawy (1997). Using such frameworks, one sound Bayesian network could be instantiated "on the fly" once a context for the value of *place* is known. However, then rules would be designed instead of a directly understandable Bayesian network. Furthermore and most importantly, *Place* is a random variable itself, and one intends to reason over it as well. *Place* represents a context-specific independence as introduced by Boutilier et al. (1996), for which frameworks such as Bayesian multinets by Geiger and Heckerman (1996) or partition-based models and contingent Bayesian networks by Milch et al. (2005) could be utilized. However, such formalisms introduce novel calculi for solving inference problems using novelly introduced operators. These novel calculi render classically known algorithms and approaches unusable, affect the intuitive and local understandability of CPDs, and may impact the computational complexity for solving classical problems. To preserve classically known properties, as well as algorithms and approaches for inference in Bayesian networks and dynamic Bayesian networks, we intend to remain a classical calculus solely using classical random variables. To do so, a naive approach would be to duplicate some random variables in the joke-cry-domain, forming a similar structure to commonly known dynamic Bayesian networks (DBNs). The latter origins from a perspective of DBNs to simulate a cyclic-feedback-behavior of Markov-random-networks by the use of DBNs resolving cycles over time.

In the following we show that cyclic dependencies also directly arise in domains directly demanding DBNs, such that one is no longer able to naively resolve cycles over time between two timesteps, as, then, a DBN would expand to infinity already between two timesteps. Moreover, we show that the limited expressiveness leads to severe problems on the soundness of inference results. Consequently, we show that the joke-cry domain and similar domains in DBNs are immediately well-defined DPGMs, called ADBNs, which overcome all of the above-mentioned issues. Moreover, we show that ADBNs preserve classically known random

variables, classical graph structures and classical CPDs, without introducing any novel calculus and without introduction of any overhead for solving commonly known inference problems in such DBNs.

3. Dynamic Bayesian Networks: Preliminaries

The following definitions, propositions and theorems on dynamic probabilistic graphical models (DPGMs) and dynamic Bayesian networks (DBNs) follow familiar notations and definitions. However, painstaking definitions are needed to shed light on marginal details, which lead to a significant problem of anticipating indirect influences in DBNs and limit the expressiveness of (D)BNs as we show in the following section.

A DBN is a DPGM that models influences between random variables and between random variables of consecutive timesteps as a Markov process repeated over time. To do so, a DPGM models dependencies between random variables where random variables exist in fixed timeslice and are dependent on random variables from the same or from previous timeslices. Random variables that depend directly or indirectly on a random variable from a previous timeslice are stateful, which is why we call them state variables.

Notation 1 (State variables)**.** *Let X_i^t be a random variable. X_i^t is the i^{th}* state variable *X_i at time t, where time t is represented by the t^{th} discrete timeslice of a model and t represents some wall-clock time t. Every X_i^t is assignable to a value $x_i \in \mathrm{dom}(X_i^t)$, where $\mathrm{dom}(X_i^t) = \mathrm{dom}(X_i)$ for all t. Let $\vec{X}^t$ be the vector of all n state variables at time t s.t.*

$$\vec{X}^t = \left(X_1^t, \ldots, X_n^t\right)^\intercal .$$

A state variable X_i^t is a random variable that bears a history, i.e., is influenced by some predecessor X_j^{t-1}. Let $P(X_i^t = x_i)$ (or $P(x_i^t)$ for brevity) denote the probability of state X_i having x_i as a value at time t. If $\mathrm{dom}(X) = \{true, false\}$ we write $+x^t$ for the event $X^t = true$ and $\neg x^t$ for $X^t = false$. If X_i^t is unspecified and not fixed by evidence, $P(X_i^t)$ denotes the probability distribution of X_i^t w.r.t. all possible values in $\mathrm{dom}(X_i)$.

To represent influences and dependencies between state variables, a DPGM is specified by one initial model and a dependency pattern between variables from consecutive timeslices. It is assumed that dependencies and influences remain constant, and a modeled pattern is repeated for multiple timeslices forming an ever expanding model. For the scope of this article we focus on models where dependencies solely exist between two consecutive timeslices (Markov-1 assumption).

Definition 1 (Dynamic probabilistic graphical model, DPGM)**.** A *DPGM* is defined as a tuple $(B_0, B_\rightarrow)$ with B_0, an initial PGM representing time $t = 0$ containing all state variables X_i^0 in $\vec{X}^0$, and $B_\rightarrow$, a consecutively repeated directed graph fragment for defining state dependencies between X_i^s and X_j^t, with $X_i^s \in \vec{X}^s, X_j^t \in \vec{X}^t, s \leq t$. For every random variable X_i^t a local CPD over all parents of X_i^t is specified. Repeating $B_\rightarrow$ for every time step $t > 0$ creates the t^{th} timeslice, by which a DPGM $(B_0, B_\rightarrow)$ is unfolded into a directed graph representing a *probabilistic graphical model* (PGM) B over random variables $\vec{X}^{0:t}$. The global semantics of B is defined as the joint probability distribution over all random variables, i.e., $P(\vec{X}^{0:t})$. The *global semantics* of a DPGM $(B_0, B_\rightarrow)$ is then defined as B's global semantics. ▲

Commonly, in a DPGM one distinguishes between permanently observed and unobservable variables (sensors and hidden states, respectively). For our work, we consider a fully observable Markov model containing only observable (but not constantly observed) random variables, i.e., hidden Markov models with varying observability spaces. Moreover, we will consider specific forms of DPGMs, namely, dynamic Bayesian networks. The relation of PGMs to Bayesian networks and of DPGMs to dynamic Bayesian networks is similar:

Proposition 1 (Dynamic Bayesian network, DBN)**.** A DPGM $(B_0, B_\rightarrow)$ is called a *DBN* and is well-defined, if state dependencies defined in $B_\rightarrow$ are limited s.t. no cyclic dependencies are created during unfolding, i.e., B is a directed acyclic graph. This is the case if $B_\rightarrow$ is a directed acyclic graph fragment. An unfolded DBN represents a Bayesian network (BN), by which the *global semantics* of $(B_0, B_\rightarrow)$ is given by the product of all locally defined CPDs of all timeslices. ▲

A proof of Proposition 1 is given as follows. With a topological ordering of all random variables in an unrolled DBN, i.e., firstly ordered temporally, then by the topological ordering of B_0 and $B_\rightarrow$, the joint probability is given due to Bayes' chain rule as a product of conditional probability distributions. Conditional independence assurances specified in B_0 and $B_\rightarrow$ then reduce all factors of the joint probability to all locally defined CPDs. It is the simplicity of a (D)BN's global semantics as the sheer product of all locally defined CPDs that allows CPDs to define their own local semantics, i.e., to provide local interpretations of probability values and specifications of (in)dependencies without considering global normalization factors. The global semantics and local semantics of (D)BNs both account for the eligibility of (D)BNs as a world-representing first-class declaration as emphasized by Pearl and Russell (2003). DPGMs, and therefore DBNs, are especially useful for reasoning about evolutions of processes and performing analyses in retrospect, e.g., weather forecasts, analyzing the current weather situation, or reconstructing events that lead to a hail storm.

In the following, we revisit roles of indirect influences and dependencies in DBNs to broaden the representation abilities of (D)BNs while maintaining sound (D)BN semantics. As mentioned afore, we consider the local semantics of DBNs as a highly valuable property, which emerges from a causal representation of dependencies in DBNs as elaborated in Section 1. However, we show that in certain domains, dependencies are not causally representable anymore in classical DBN formalisms and one loses the ability to anticipate indirect influences. We show that in such domains, cyclic dependency structures are evident from causal and local views on dependencies, and that cyclic dependency structures are required in order to provide an intuitive parametrization of such models and to anticipate all indirect influences. By reconsidering the roles of dependencies, we show that such DBNs can be based on cyclic graphs while preserving local semantics, preserving a classical calculus with classical random variables and preserving classically known algorithms for DBNs without introducing any external frameworks. To do so, we differentiate between different structural forms of DPGMs and DBNs.

Notation 2 (Inter- and intra-timeslice dependencies, and diagonal models)**.** *For state dependencies defined in $B_\rightarrow$ of the form $t-1 \leq s \leq t$ (see Definition 1), one speaks of a* first-order Markov property*, which we focus on in this article. For any DPGM with $t-1 \leq s < t$, i.e., states at time t are only dependent of states at time $t-1$, an acyclicity constraint in the directed graph holds and a DPGM is a well-defined DBN. We call dependencies of the form*

$t-1 \leq s < t$ inter-timeslice *dependencies. We call DPGMs in which only inter-timeslices dependencies are defined* diagonal *models. For a limited set of dependencies of the form* $t-1 \leq s \leq t$*, called* intra-timeslice *dependencies, a DPGM is a well-defined DBN, as long as no directed cycles are created.*

More often than not, many DBNs are designed as diagonal models (as in Figure 3, gray), and are introduced due to syntactic constraints on (D)BNs, but stand in conflict with an actual causality in their domain, as the following examples demonstrate. Such dependencies exist causally at $s = t$, but would create directed cycles in one timeslice. Dependencies on "sibling" states of one timeslice are then "spread over the past" and conflict with causality. This means that indirect causes among siblings are not anticipated correctly in a particular state, such that "chain reactions" are not covered appropriately. The following example introduces a running example used throughout this article.

Example 2 (Regulatory compliance)**.** *In a company a corrupt employee deliberately places fraudulent information, e.g., faked payment sums for bribe money, which divulge throughout a company until every employee has (unknowingly) committed a compliance violation, i.e., has become corrupt, too. In order to trace back a potential source for a detected compliance violation, to reconstruct sequences of events and to prevent future compliance violations, we model a probabilistic regulatory-compliance domain using a DBN. If one employee spreads fraudulent information, we do not say that such an employee is "corrupt," but in this context we assume that he is* tainted*. We speak of taintedness, because saying that an employee is corrupt implies that he is knowingly manipulating information; being tainted shall represent that he might distribute fraudulent information indeliberately. The taintedness state of every employee,* Claire*,* Don *and* Earl*, say, is represented by a state random variable in* $\vec{X}^t$*. The probability* $P(X_i^t)$*, encodes the belief in employee* X_i *being tainted* $+x_i^t$ *or being integrous* $\neg x_i^t$ *at time* t*. We model* B_0 *s.t. it models our prior belief in every employee being a source of fraudulent information, i.e.,* B_0 *is a BN containing all* $\vec{X}^0$ *as random variables without any influences; say* $P(+c^0) = 0.5, P(+d^0) = 0.6, P(+e^0) = 0.7$*. Being tainted is assumed to be permanent, such that* $B_{\rightarrow}$ *describes all random variables* X_i^t *depending on* X_i^{t-1} *with conditional probability* $P(+x_i^t | +x_i^{t-1}) = 1$*.*

An employee might influence another employee in his writings or, rather, in his information state. A tainted employee might therefore (indeliberately) influence his colleague such that the colleague also falls prey to fraudulent information, i.e., becomes tainted, too. Influences happen through message exchanges, i.e., only if a message is passed from employee X *to* Y *at* t *an influence is exerted. We represent message exchanges by random variables* M_{XY}^t *as part of our domain, with* $+m_{XY}^t$ *indicating that* Y *receives a message from* X *at time* t*.*

In the example one now can make observations, e.g., from unheralded compliance checkups, and trace a potential diffusion of false information throughout our company over time. Say, Claire influences Don, and if Claire is tainted there is a probability of Don becoming tainted, too. Further, if Don influences Earl there is a probability that Claire influences Earl *indirectly* through Don, i.e., Claire is an indirect cause of Earl becoming tainted. If one is inclined to model only this dependency of E on D on C, one can correctly represent the domain as in Figure 2. In this minimal example, indirect influences occur and are correctly anticipated. However, if more potential influences are supposed to be modeled one is at odds with causality as the following will show.

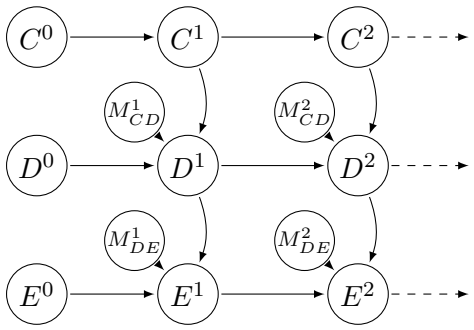

Figure 2: A causally correctly represented world for the minimal case of Example 2 using a DBN. Messages M^t_{XY} are only possibly sent from *C*laire to *D*on and from *D*on to *E*arl. Modeling more possible message exchanges, i.e., influences, leads to Figure 3.

Considering Example 2, we intend to model that *all employees can potentially influence each other*, which, however, would lead to a cyclic graph structure in $B_{\rightarrow}$ of a DPGM (as seen in Figure 3, black). In order to obtain a well-defined DBN, one must "bend" some dependencies to a consecutive/previous timestep in $B_{\rightarrow}$ (as done in Figure 3, gray) forming a diagonal model. This is unavoidable, but is seen as inaccurate from a world representation point of view, as indirect influences are now anticipated spuriously. Bending an influence to a consecutive timestep encodes an incubation time and, essentially, models a different process: Earl is now influenced by Claire *through* Don from a Claire of the penultimate timepoint. To approximate the intended meaning, a timeslice must be infinitesimally small to somehow anticipate all indirect influences and cannot be chosen appropriately for an intended use case, e.g., to represent a daily acquisition of information.

In fact, a cyclic taintedness domain is directly evident in a cyber security application, assessing how locally inflicted adverse effects on devices or services (taintedness of employees) in an ICT network "spread" throughout the network by datatransfers (message transfers, $\vec{b}$) leading to causal chain of events potentially striking highly critical devices. A source of such local adverse effects are, e.g., vulnerabilities, which represent a probability that a device may be threatened by an adversary, i.e., prior belief probabilities in B_0, or are, e.g., alarms raised by intrusion detection systems or antivirus applications at time t representing direct observations $\vec{z}^t$. Every received data item from such impacted device bears a probability that data was compromised or exfiltrated leading to a spread of such impacts throughout a network. If highly critical devices are potentially not operating as intended, a complete company might fail their business goals, or missions are not accomplishable anymore. Projecting these local impacts globally onto a higher goal, such as a company or a mission, while considering all indirect and transitive effects is frequently called a *mission impact assessment*. Motzek and Möller (2017) discuss an extension of Motzek et al. (2015) for a probabilistically sound mission impact assessments extended towards dynamic domains based on cyclic ADBNs. A cyclic ADBN structure closely resembling the taintedness domain immediately arises. As observations of datatransfers are acquired from different locations inside a network, imprecise timestamps must be synchronized, leading to a potentially unknown temporal ordering of observations, i.e., coarse time slices must be chosen. Moreover, coarser time slices are beneficial for the computational demands of performing inference. Such dynamic mission

impact assessments are then used to analyze the current situation, i.e., to raise situational awareness, and to perform forensic analysis in retrospect to discover chains of events that led to a potential compromise or impact onto a company. In fact, (A)DBNs are predestined for such an application: every parameter in a dynamic mission impact model can be validated and parametrized intuitively, as every parameter is a locally understandable conditional probability distribution. Without such a parameter-validatability, multiple complete impact assessments, e.g., observations from sources of impacts to effects on a global business, are required to train and validate a model from and against ground truth; but the acquisition of such ground truth quickly becomes impossible, as thousands of such observations with actual business impacts are required, which would ruin the business per se while acquiring an initial dataset. Fortunately, when using a validated ADBN, commonly known inference problems such as filtering and smoothing (cf. Section 5), then provide dynamic mission impact assessments, which are immediately validated based on the validated parameters. As an obtained impact assessment is defined to be correct and is a plain conditional probability, the assessment is directly understandable by every person and does neither require reference results for comparison, nor a deep understanding of the underlying mathematical principles, nor any security knowledge background. As motivated by the taintedness domains, non-infinitesimal, i.e., coarse, timeslices demand cyclic dependency structures, which are, however, not well-defined in classical DBN formalisms. In the following, we show that bending of dependencies to a consecutive timestep forming diagonal models leads to severe issues, delivering highly spurious results. Further, we show that the cyclic dependencies in the taintedness domain represent a novel, well-defined DPGM which preserves the desired local semantics without introducing a novel calculus nor any computational overhead for performing inference.

Remark 1 (Use of diagonal models)**.** *Diagonal models can be used to simulate a cyclic "feedback" relationship by letting a model converge to a stable state after long periods of time, which is used to simulate hidden Markov models with the use of DBNs, as, e.g., done by Murphy (2002) or Ghahramani (2001). However, when seeing each timeslice as a representation of a fraction of a real "wall-clock" time, as done by, e.g., Glesner and Koller (1995), Sanghai et al. (2005), Jaeger (2001), and us, a cyclic model would already expand to infinity during one timeslice, e.g., when going from day* 1 *(*$t = 1$*) to day* 2 *(*$t = 2$*).*

In the following we show that without a causal design, i.e., without modeling direct and indirect causes correctly, spurious results are obtained from such models. We further show that by allowing cyclic dependency structures, one anticipates indirect influences without any demands on time granularity in the form of a novel DPGM class, called ADBNs. Moreover, we show that ADBNs are remarkable similar to DBNs, but without an acyclicity constraint.

Classically, a conditional independency in a Bayesian network is represented by the lack of an arc between two nodes. Another kind of independence in Bayesian networks, called context-specific independence (CSI), has been studied by Boutilier et al. (1996) and Ngo and Haddawy (1997). CSIs represent dependencies in a BN that are only present in specific contexts and have mainly been used for more efficient inference, e.g., as studied by Poole and Zhang (2003) (cf. Section 8). We use the notation A_{XY} if a random variable acts as a so called *activator* random variable which activates a dependency of random variable Y on X in a given context.

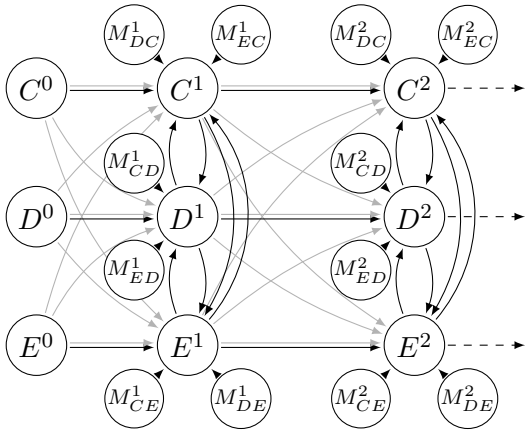

Figure 3: Two options (black/gray) to represent a world using an (A)DBN for the running example if every employee (*C*laire, *D*on, *E*arl) can influence every other employee through messages M^t_{XY}. Syntactic DAG constraints of (D)BNs prevent cyclic dependencies, and diagonal state dependencies are enforced (indicated with gray arrows). The diagonal option is considered inaccurate from a world representation point of view. In the diagonal case, M^t_{XY} represents $M^{t-1\,t}_{XY}$, i.e. M^t_{XY} affects the dependency of state Y^t on X^{t-1}.

Definition 2 (Activation / deactivation criteria)**.** Let $\mathrm{dom}(A_{XY}) = \{true, false\}$ (extensions to non-boolean domains are straightforward). The *deactivation* criterion for $A_{XY} = false$ is defined as

$$\begin{aligned} &\forall x, x' \in \mathrm{dom}(X), \forall y \in \mathrm{dom}(Y), \forall \vec{z} \in \mathrm{dom}(\vec{Z}) : \\ &\qquad P(y|x, \neg a_{XY}, \vec{z}) = P(y|x', \neg a_{XY}, \vec{z}) = P(y|*, \neg a_{XY}, \vec{z}) \ , \end{aligned} \tag{1}$$

where $*$ represents a wildcard and $\vec{Z}$ remaining direct dependencies of Y, i.e., direct parents of Y. Given $\neg a_{XY}$, we say that a direct causal dependence of Y on X, i.e., a direct causal influence of X on Y, is *inactive*.

The *activation* criterion describes a situation where Y becomes directly dependent on X, where the CPD entry for some $y \in \mathrm{dom}(Y)$ is not uniquely identified by just $+a_{XY}$ and $\vec{z}$, hence

$$\begin{aligned} &\exists x, x' \in \mathrm{dom}(X), \exists y \in \mathrm{dom}(Y), \exists \vec{z} \in \mathrm{dom}(\vec{Z}) : \\ &\qquad P(y|x, +a_{XY}, \vec{z}) \neq P(y|x', +a_{XY}, \vec{z}) \ . \end{aligned} \tag{2}$$

Given $+a_{XY}$, we say that the direct causal dependence of Y on X, i.e., the direct causal influence of X on Y, is *active*.

If for a random variable A^t_{XY} both activation and deactivation criteria are fulfilled by a local CPD definition of random variable Y, A^t_{XY} is called an *activator* random variable. ▲

Note that Definition 2 is based on properties of locally defined CPDs of random variables in a (D)BN, i.e., some random variable of a (D)BN is identified to be an activator random

variable by some specific numerical parameter settings in CPDs. The activation criteria and deactivation criteria only apply to these local CPDs, and do not apply to the global semantics of a (D)BN. Table 1 shows an arbitrarily specified CPT in which random variables show to have an activator nature and follow Definition 2. Note also that activator criteria can be present in any form of CPDs, as expressed in the following remark.

Remark 2. *The (de)activation criterion can be summarized as: a probability is uniquely identified by active dependencies, and inactive dependencies become irrelevant. This property is easily confusable with a property of a "noisy-or combination" function (e.g., described by Henrion, 1988), where "false" dependencies (or as Heckerman and Breese (1996) say: in a "distinguished" state) are semantically supposed to be irrelevant. Under noisy-or assumptions, each conditional probability value in a CPD is obtained by a deterministic combination function of individual probability fragments associated with "true" dependencies. Still, the general activation and deactivation criteria from Definition 2 are* not *linked to specific combination functions and can be present in arbitrarily specified CPDs (cf. Table 1). Nevertheless, the noisy-or combination function inherently defines activation and deactivation criteria in their semantics, where only "true dependencies act as potential causes" and a "false dependence does not cause any harm." Motzek and Möller (2015a) call such a property an "innocuousness-property," and explore its role in ADBNs, as ADBNs actually enable one to (semantically and syntactically) formalize such a property in CPDs. The difference between the innocuousness property and the activator criterion is that for the general (de)activation criterion the presence (the activation) of a dependence is* relevant *for every value. For innocuousness properties of CPDs, the presence of a "false" dependence is* irrelevant*, i.e., the activation is only of interest for some values of the dependence.*

Furthermore, the Example 3 below demonstrates the identification of activator random variables on the running example of the taintedness domain.

Example 3 (Regulatory compliance continued)**.** *In Example 2 we modeled that Claire does not constantly exert an influence on Don, i.e.,* only if *Claire sends a letter to Don there is an influence made explicit. In fact, message exchange variables M^t_{XY} from Example 2 act as activator random variables according to Definition 2.*

We observe possible message exchanges from utilized envelopes (possibly found in the trash bin). On internal envelopes one usually finds indicators for multiple transfers from a coarse time interval in an imprecise or inaccurate order. For example, a transfer from Don to Earl and one from Claire to Don might include a transitive influence of Claire on Earl during the same time interval. *We show in the following that it is highly important to cover these indirect influences and to model indirect causes appropriately.*

Example 3 shows that activator random variables naturally exist in domains and do not necessarily need to be introduced as auxiliary variables. If activator random variables are present in domains, an (effective) structure of an DPGM is not known in advance and even changes over time. As effective structures are not known in advance, only general structures are designable in advance, covering all potential substructures. To correctly cover all implications of influences, i.e., to consider all direct and indirect causes, generally cyclic DPGMs are required (as in Figure 3, black).

Table 1: Example for a CPD $P(+x|V,W,Y,Z)$ with arbitrary numbers α—γ. In fact, random variables V and W are identifiable as activator random variables according to Definition 2. V represents an A_{YX} and W represents an A_{ZX}.

V	**W**	**Y**	**Z**	**P**$(+x|\dots)$	**V**	**W**	**Y**	**Z**	**P**$(+x|\dots)$
$+v$	$+w$	$+y$	$+z$	α	$+v$	$\neg w$	$\neg y$	$+z$	ν
$+v$	$\neg w$	$+y$	$+z$	β	$+v$	$\neg w$	$\neg y$	$\neg z$	ν
$+v$	$\neg w$	$+y$	$\neg z$	β	$\neg v$	$\neg w$	$+y$	$+z$	ϕ
$\neg v$	$+w$	$+y$	$+z$	ϵ	$\neg v$	$\neg w$	$+y$	$\neg z$	ϕ
$\neg v$	$+w$	$\neg y$	$+z$	ϵ	$\neg v$	$\neg w$	$\neg y$	$+z$	ϕ
$+v$	$+w$	$\neg y$	$+z$	η	$\neg v$	$\neg w$	$\neg y$	$\neg z$	ϕ
$+v$	$+w$	$+y$	$\neg z$	γ	$\neg v$	$+w$	$+y$	$\neg z$	ψ
$+v$	$+w$	$\neg y$	$\neg z$	μ	$\neg v$	$+w$	$\neg y$	$\neg z$	ψ

4. Activator Dynamic Bayesian Networks

A DPGM in which some random variables are identifiable as activator random variables according to Definition 2 is called an Activator DBN, i.e., random variables in B_0 and $B_\rightarrow$ can syntactically be grouped into two (not necessary disjoint) sets of state variables $\vec{X}^t$ and activator variables $\vec{A}^t$.

Notation 3 (Activator matrices)**.** *Let $A_{ij}^{s\,t}$ be the activator random variable influencing X_j^t regarding a dependency on X_i^s. Let $A^{s\,t}$ describe the* matrix of all activator random variables *between timeslice s and t s.t.*

$$A^{s\,t} = \begin{pmatrix} A_{11}^{s\,t} & \cdots & A_{1n}^{s\,t} \\ \vdots & \ddots & \vdots \\ A_{n1}^{s\,t} & \cdots & A_{nn}^{s\,t} \end{pmatrix} .$$

Let $\vec{A}_i^{s\,t}$ denote the i^{th} column of $A^{s\,t}$, i.e., $\vec{A}_i^{s\,t}$ represents the vector of all activator random variables *relevant for X_i^t regarding an influence by random variables of timeslice s. Let $\vec{\mathcal{A}}^{s\,t}$ denote the corresponding column vector of all entries of $A^{s\,t}$, i.e.,*

$$\vec{\mathcal{A}}^{s\,t} = \left(A_{11}^{s\,t}, \dots, A_{1n}^{s\,t}, \dots, A_{n1}^{s\,t}, \dots, A_{nn}^{s\,t}\right)^\intercal .$$

Let $\vec{\mathcal{A}}^{0:tt} = \langle \vec{\mathcal{A}}^{01}, \vec{\mathcal{A}}^{11}, \vec{\mathcal{A}}^{12}, \dots, \vec{\mathcal{A}}^{tt} \rangle$ denote the vector of all activator random variables existing in and between timeslices 0 to t. For brevity, we write A^t for A^{tt} (excluding A_{kk}^{tt}), and correspondingly we write A_{ij}^t, $\vec{A}_i^t$ and $\vec{\mathcal{A}}^t$. Let $\vec{\mathcal{A}}^{1:t} = \langle \vec{\mathcal{A}}^1, \vec{\mathcal{A}}^2, \dots, \vec{\mathcal{A}}^t \rangle$ then denote the vector of all intra-timeslice activator random variables $\vec{\mathcal{A}}^t$ for every timeslice t.

Activator dynamic Bayesian networks are DPGMs and share familiar syntax and semantics with dynamic Bayesian networks (compare Definition 1 and Proposition 1), but ADBNs are not bound to DAG constraints like DBNs:

Definition 3 (Activator dynamic Bayesian network, ADBN)**.** An *ADBN* is syntactically defined as a tuple $(B_0, B_\rightarrow)$ with B_0 defining an *initial Bayesian network* representing time $t = 0$ containing all states $X_i^0 \in \vec{X}^0$, and a consecutively *repeated activator Bayesian network fragment* $B_\rightarrow$ consisting of dependencies between state variables X_i^s and X_j^t, $t-1 \leq s \leq t$, (Markov-1) and consisting of dependencies between state variables X_i^t and activator random variables A_{ji}^{st}. For every random variable X_i^t, A_{ij}^{st} a local CPD over all parents, e.g., as a CPT is specified, where CPDs of state variables X_i^t follow Definition 2.

By repeating $B_\rightarrow$ for every time step $t > 0$, an ADBN $(B_0, B_\rightarrow)$ is unfolded into a PGM defining an *ADBN's global semantics*. ▲

Definition 3 defines ADBNs as a form of DPGM in which some random variables have an activator nature, i.e., some CPDs of state variables show certain properties that identify other random variables as activator random variables. Note that these activator random variables are *not* introduced externally, but are already part of a domain as shown in the following examples. The following theorem states in which cases an ADBN is well-defined. The idea is that two cases may exist: (1) an ADBN is a classical DBN, i.e., acyclic and well-defined by Proposition 1, or (2) an ADBN is not a classical DBN, i.e., it contains cyclic dependencies, for which a different well-definedness condition is introduced, which is a modified constraint of acyclicity.

Theorem 1 (ADBN well-definedness)**.** An ADBN is well-defined, if an ADBN is well-defined according to Proposition 1, i.e., if it is a well-defined DBN. An ADBN is well-defined for every instantiation $\vec{a}_*^{1:t}$ of $\vec{\mathcal{A}}^{1:t}$, if for all t, $\vec{a}_*^t$ satisfies the acyclicity predicate $\mathfrak{A}$:

$$\begin{aligned} \forall x, y, z \in \vec{X}^t : \mathfrak{A}(x,z)^t, \mathfrak{A}(z,y)^t \rightarrow \mathfrak{A}(x,y)^t \\ \neg \exists q : \mathfrak{A}(q,q)^t \ , \end{aligned} \tag{3}$$

with an acyclicity predicate $\mathfrak{A}(i,j)^t$ that is defined as

$$\mathfrak{A}(i,j)^t = \begin{cases} \mathit{false} & \text{if} \quad \neg a_{ij}^t \in \vec{a}_*^t \\ \mathit{true} & \text{if} \quad \text{else} \end{cases} .$$

For every well-defined ADBN, the global semantics, i.e., $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{01:tt^\intercal})$, is, equivalently to DBNs, given as the product of all locally defined CPDs. ▲

Theorem 1 means that ADBNs are not syntactically bound to DAG structures in $B_\rightarrow$ and still share familiar DBN semantics providing a local interpretation and specification of conditional probability distributions without the need for global normalization factors as, e.g., in conditional random fields. A proof for Theorem 1 is later given in the appendix by Proof A. Well-definedness is achieved through the novel acyclicity constraint, namely that not a syntactic graph structure is forced to be acyclic, but rather that an *instantiation* of an unrolled DBN is acyclic. This is useful for applications where actual structures are not known in advance and might change at every timeslice depending on specific contexts. Such particular situations often arise, if timeslices are not supposed to be infinitesimally small, and indirect influences must indeed be considered, with cyclic dependency structures in $B_\rightarrow$ arising naturally during a design phase. In particular, the novel acyclicity constraint

of ADBNs is highly beneficial for the running example, as demonstrated in Example 4 after the following definitions.

The novel acyclicity constraint means that it has to be enforced that only well-defined instantiations are used for inference. Enforcing the use of only well-defined instantiations can be ensured by adequate observations or modeling approaches (see later Section 7). We characterize such well-defined instantiations using the following definition.

Definition 4 (Regular and acyclic instantiations)**.** If an instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ of $\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t}$ leads to a well-defined ADBN according to Theorem 1, we say that the instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ is *regular*. Let G_A^t represent the graph formed by active activators of a full instantiation $\vec{a}^t$, where every active activator $+a_{ij}^t \in \vec{a}^t$ represents an edge from node i to j. If, for every $\vec{a}^i$ in $\vec{a}^{1:t}$, G_A^i is acyclic, we say that the instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ is *acyclic*. ▲

Under Definition 4 every acyclic instantiation is regular. For the predicate $\mathfrak{A}$ defined in Theorem 1 every regular instantiation is also an acyclic instantiation. Note that Motzek and Möller (2015a) show that $\mathfrak{A}$ predicates exist for which regular instantiations do not necessarily need to be acyclic, i.e., not only contexts of activator random variables can ensure regularity and it need not be the goal to ensure acyclicity, but it is the goal to ensure regularity.

Corresponding to Notation 2 regarding inter- and intra-timeslice models, we distinguish ADBNs based on the density of their activator matrices:

Notation 4 (Dense, inter- and intra-timeslice ADBNs)**.** *An* intra-timeslice *ADBN* $(B_0, B_\rightarrow)$ *is an ADBN with non-empty activator matrix* $A^{t\,t}$*. A* diagonal *or* inter-timeslice *ADBN* $(B_0, B_\rightarrow)$ *is an ADBN with non-empty* $A^{t-1\,t}$*. An ADBN* $(B_0, B_\rightarrow)$ *is called a* dense *ADBN, if at least one activator matrix* $A^{s\,t}$ *is dense, i.e., there exists a random variable acting as an activator random variable for every modeled influence. In a dense intra-timeslice model, cyclic dependencies exist in* $B_\rightarrow$*, for which we also speak of* cyclic *ADBNs.*

Note again that activators in an ADBN are classical random variables and are part of a modeled domain as shown in the following example, i.e., activators are not auxiliary variables.

Example 4 (Inter- and intra-timeslice ADBNs)**.** *Based on Example 3 the regulatory compliance domain can be modeled as an ADBN: Message exchange variables have activator nature and one obtains an ADBN with activators* $\vec{\mathcal{A}}^t = (M_{CD}^t, M_{DC}^t, M_{DE}^t, M_{ED}^t, M_{CE}^t, M_{EC}^t)^\intercal$ *and state random variables* $\vec{X}^t = (C^t, D^t, E^t)^\intercal$*.*

To model that every employee can potentially influence every other*, i.e., can send him a document, two options are available: (a) a diagonal (dense) inter-timeslice ADBN as shown in Figure 3 (gray), i.e., a well-defined DBN according to Proposition 1, or (b) a cyclic (dense) intra-timeslice ADBN as shown in Figure 3 (black), i.e., a well-defined ADBN according to Theorem 1, if correctly instantiated.*

As discussed earlier, this example highlights both design options for the discussed taintedness domain: one classic solution based on a diagonal ADBN and one using a newly allowed cyclic ADBN. As a diagonal (dense inter-timeslice) ADBN represents a classic DBN, we also write (A)DBN in the diagonal case. Note that the diagonal (A)DBN solution is the

only consequent well-defined classic DBN solution[1]. However, as discussed earlier, a diagonal option is seen as *inaccurate* from a world representation point of view and is subject to a significant problem with indirect causes as punctuated in the following example.

Example 5 (Diagonal (A)DBN restrictions example)**.** *Continuing Example 4, say, one observes a message transfer from Claire to Don (*$+m^1_{CD}$*) and from Don to Earl (*$+m^1_{DE}$*), and one can neglect all other transfers, i.e.,* $\neg m^1_{DC}, \neg m^1_{ED}, \neg m^1_{CE}, \neg m^1_{EC}$*. To fully evaluate all implications of the observations, one has to anticipate an indirect influence from Claire to Earl through Don during timeslice* 1*. But, the diagonal (A)DBN in Figure 3 (gray) does not encode the domain correctly and leads to spurious results: Earl is only influencable by Claire* through *Don from a Claire of the penultimate timepoint, but the observed message exchanges only let* t_1*-Claire influence* t_2*-Don, and let* t_1*-Don influence* t_2*-Earl. Cf. later Examples 7 and 8.*

Example 5 emphazises that the diagonal (A)DBN option encodes a different domain where influences between random variables only affect a consecutive timeslice, i.e., diagonal (A)DBNs represent an incubation time, which implies that a time-granularity must be adequately small. However, if information is only acquirable at a daily or weekly scale, incubation times elapse, and one expects consecutive influences to be considered during the same timeslice, i.e., one expects the anticipation of indirect influences, which cannot be represented in a classical diagonal (A)DBN. In effect, one is limited to reasoning using only direct influences in diagonal (A)DBNs as formalized in the following proposition.

Proposition 2 (Diagonal (A)DBN restrictions)**.** Let $(B_0, B_\rightarrow)$ be a diagonal inter-timeslice (A)DBN and let $\vec{a_*}^{1:t}$ be an instantiation of $\vec{\mathcal{A}}^{1:t}$ for which holds $\exists i, j, k, t : {+}a^t_{ij} \in a^t_* \wedge {+}a^t_{jk} \in a^t_*$. Then, $\vec{a_*}^{1:t}$ includes an indirect influence and *enforces an infinitesimal resolution* of timeslices, where indirect effects do not need to be anticipated. If a resolution of timeslices is not infinitesimally small, inference results from $(B_0, B_\rightarrow)$ under $\vec{a_*}^{1:t}$ will be *spurious*. Only if $\forall t \neq \exists i, j, k, t : {+}a^t_{ij} \in a^t_* \wedge a^t_{jk} \in a_* t$ holds for $\vec{a_*}^{1:t}$, $\vec{a_*}^{1:t}$ is *indirect-free* and supported by a diagonal (A)DBN. ▲

Proposition 2 shows that diagonal (A)DBNs are severely restricted and may return spurious results if a time granularity is not chosen appropriately for the model (as it has been the case in Example 5). If time granularity is not chosen appropriately according to Proposition 2, inference results may become spurious since evident indirect influences cannot be anticipated. Effectively, one is only able to reason over instantiations of a diagonal (A)DBN where the instantiations of activators form a bipartite digraph with uniformly directed edges, i.e., only direct influences occur. The latter significantly limits the expressiveness and reasoning abilities of diagonal DBNs. The necessity to adjust the resolution of time to properties of the chosen framework, instead of adjusting the resolution to the modeled domain, stands in significant conflict with our intention of maintaining a Bayesian network as a first-class representation of the world.

1. A limited set of dependencies can be modeled as intra-timeslice dependencies, but it is an arbitrary decision why some dependencies are modeled as inter- and some are modeled as intra-timeslice dependencies. Moreover, an arbitrary choice leads to severe parametrization issues as discussed in Section 2.

Fortunately, Theorem 1 explicitly permits structures of $B_{\rightarrow}$ that are forbidden under classic DBN definitions: cyclic intra-timeslice models, as discussed in Example 4, are indeed well-defined under some restrictions according to Theorem 1. In the following we show that restrictions of intra-timeslice models are less strict than restrictions on classic, diagonal (A)DBN models:

Example 6 (Cyclic (A)DBN restrictions example)**.** *Continuing Ex. 5, with the same message transfer observations, but for a cyclic ADBN option (Figure 3, black): The observations only allow regular and acyclic instantiations which satisfy Theorem 1 and thus lead to a well-defined Bayesian network, even though $B_{\rightarrow}$ is based on a cyclic graph. Here, all implications of the observations, namely an indirect influence from Claire to Earl through Don during timeslice* 1 *is anticipated. As intended, t_1-Claire does influence t_1-Don of the same timeslice, and t_1-Don influences t_1-Earl of the same timeslice. To achieve this constellation in a general setting, $B_{\rightarrow}$ must proactively be designed with cycles, as actual dependencies are not known in advance and change over time.*

This example shows that in ADBNs indirect influences are anticipated under a time-granularity suited to a problem (e.g., days) instead of a time-granularity enforced by a reasoning framework. Essentially in some sense, ADBNs move an acyclicity constraint for well-definedness from the design phase of a Bayesian network to an actual instantiation of a Bayesian network at "runtime."

To summarize, we have shown that classic DBNs are limited in their expressiveness, and instantiations of them are limited to indirect-free instantiation sets that do not contain indirect influences if time granularity is not infinitesimally small. With the identification of activator properties of random variables in ADBNs, more expressive graph structures for (A)DBNs are supported, while sharing similar semantics, and with larger sets of regular instantiations of ADBNs being possible. Later, in Section 8, we quantitatively compare restrictions on the number of regular instantiations between DBNs and ADBNs.

In fact, every possible Markov-1 DBN can be represented as a dense inter-and-intra timeslice ADBN as the next proposition states.

Proposition 3 (Completeness)**.** An ADBN can model any joint probability and includes every possible Markov-1 DBN structure. Let $(B_0^*, B_{\rightarrow}^*)$ be a dense inter- and intra-timeslice ADBN with a dense inter-timeslice activator matrix $A^{t-1\,t}$ and a dense intra-timeslice activator matrix $A^{t\,t}$. Then, for every Markov-1 DBN $(B_0', B_{\rightarrow}')$ there exists a minimal set of instantiations $\vec{a}_{\blacklozenge}$ of $(B_0^*, B_{\rightarrow}^*)$ under which the same effective topological ordering is formed as defined by $(B_0', B_{\rightarrow}')$. Therefore, a dense inter-timeslice ADBN represents a superclass of all possible inter-timeslice DBN structures and a dense intra-timeslice ADBN represents a superclass of all possible intra-timeslice DBN structures. ▲

We argue that often DBN models with diagonal state dependencies are used only due to syntactic constraints on (D)BNs (Proposition 1), but stand in conflict with an actual causality in their domain. Therefore, we focus in the following on a dense intra-timeslice ADBN, which covers and represents all possible novelly allowed cases of ADBNs. The following proposition derives the semantics according to Theorem 1 as the joint probability distribution over all random variables.

Proposition 4 (Joint probability distribution of a dense intra-timeslice ADBN)**.** If an ADBN is well-defined, the joint probability over all random variables is defined as the product of all locally defined CPDs. Therefore, a dense intra-timeslice ADBN's semantics is

$$\begin{aligned} P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) &= P(X_1^0) \cdot \ldots \cdot P(X_n^0) \cdot \prod_{i=1}^{t} P(X_1^i | X_2^i, \ldots, X_n^i, A_{21}^i, \ldots, A_{n1}^i, X_1^{i-1}) \\ &\cdot \ldots \cdot P(X_n^i | X_1^i, \ldots, X_{n-1}^i, A_{1n}^i, \ldots, A_{(n-1)n}^i, X_n^{i-1}) \cdot P(A_{12}^i) \cdot \ldots \cdot P(A_{n(n-1)}^i) \\ &= \prod_{X_k^0 \in \vec{X}^0} P(X_k^0) \cdot \prod_{i=1}^{t} \prod_{X_k^i \in \vec{X}^{\,i}} P(X_k^i | \vec{X}^{i\intercal} \setminus X_k^i, \vec{A}_k^{i\intercal}, X_k^{i-1}) \cdot \prod_{A_{cv}^i \in \vec{\mathcal{A}}^i} P(A_{cv}^i) \ . \end{aligned} \quad (4)$$

As expected, the joint probability can be defined recursively:

$$\begin{aligned} &P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) = \\ &\qquad P(\vec{X}^{0:t-1\intercal}, \vec{\mathcal{A}}^{1:t-1\intercal}) \cdot \prod_{X_k^t \in \vec{X}^{\,t}} P(X_k^t | \vec{X}^{t\intercal} \setminus X_k^t, \vec{A}_k^{t\intercal}, X_k^{t-1}) \cdot \prod_{A_{cv}^t \in \vec{\mathcal{A}}^t} P(A_{cv}^t) \ . \quad \blacktriangle \end{aligned} \quad (5)$$

Naively, every query to an (A)DBN can be answered by marginalization from the defined joint probability distribution over all random variables from all timeslices, which, however, is computationally intractable. Therefore, the following sections discuss special types of queries and derive exact and approximate solutions to associated problems in cyclic (A)DBNs.

5. Common Queries and Associated Answering Problems in ADBNs

A DBN is a temporal probabilistic knowledge base with respect to which queries can be posed. Common query types are known as *filtering*, *smoothing* and *most likely explanation* (we adapt the meaning of Murphy, 2002) and define query answering problems w.r.t. the semantics of the knowledge base. Queries are used to investigate historical information in retrospect or are used to constantly monitor a specific variable over time, e.g., estimate a trajectory of a moving object for which position measurement values are noisy.

As explained above, every query is answered by straight marginalization from the full joint probability distribution (JPD) defined by the unrolled DBN, i.e., from the JPD defined by a Bayesian network consisting of all timeslices upto timepoint t at once. However, such a naive approach is only tractable for the very first timeslices and the curse of dimensionality prevents inference over long periods of time. Still, it is a highly important property of DBNs that commonly known problems, such as filtering- and smoothing-problems, remain practically solvable even over long periods of time via commonly known algorithms such as the forward-backward algorithm. Without restriction of any kind, this property must be preserved in any novel dynamic probabilistic model, such as the one introduced by us—ADBNs. Therefore, we prove in this section that inference in ADBNs remains in the same complexity class as corresponding problems in multiply connected DBNs. In general, inference problems in multiply-connected (D)BNs are known to be NP-hard as shown by Cooper (1990). However, focusing on single input parameters of inference problems associated with DBNs, problems show to be fixed-parameter tractable. This means that algorithms

exist, e.g., the forward-backward-algorithm, which provide exact solutions to common query answering problems in a tractable time- and memory-complexity over time, i.e., remain tractable even for large numbers of consecutive timeslices. In this section we show and prove that inference in ADBNs does not introduce any overhead for solving commonly known problems, that solutions, such as the forward-backward-algorithm remain directly applicable for ADBNs based on cyclic graphs, and that inference in ADBNs remains solvable with the classical and familiar calculus from (D)BNs without any introduction of novel operators, i.e., one preserves classical marginalization from a JPD based on classical random variables with associated CPDs.

Definition 5 (Query answering language and observations)**.** Let $\vec{Z}^t \subseteq \vec{X}^t$ be a set of *observed* and $\vec{\zeta}^t = \vec{X}^t \backslash \vec{Z}^t$ be the corresponding set of *unobserved state variables*. Let $B^t \subseteq A^t$ be a set of *observed activators* and $\vec{\mathcal{B}}^t \subseteq \vec{\mathcal{A}}^t$ be the corresponding column vector representation. Likewise, let $\vec{\beta}^t = \vec{\mathcal{A}}^t \backslash \vec{\mathcal{B}}^t$ be the column vector of all *unobserved activators*. Then, *observations of state variables* $\vec{z}^t$ are instantiation assignments $X_i^t = x_i \in \mathrm{dom}(X_i^t)$ and *observations of activator random variables* $\vec{\mathfrak{b}}^t$ are instantiation assignments $A_{ij}^t = a_{ij} \in \mathrm{dom}(A_{ij}^t)$.

Let $\vec{x}^t \in \mathrm{dom}(\vec{X}^t)$ be a full instantiation of $\vec{X}^t$ and let $\vec{a}^t \in \mathrm{dom}(\vec{\mathcal{A}}^t)$ be a full instantiation of $\vec{\mathcal{A}}^t$. Further, let every instantiation assignment in $\vec{x}^t, \vec{z}^t, \vec{a}^t, \vec{\mathfrak{b}}^t$ uniquely define the value of its respective random variable in $\vec{X}^t$ or $\vec{\mathcal{A}}^t$.

Then, a *query* $P(\vec{x}^{k\intercal}, \vec{a}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathfrak{b}}^{1:t\intercal})$ to a probabilistic knowledge base $(B_0,\ B_{\rightarrow})$ is a request for the respective result of $P(\vec{x}^{k\intercal}, \vec{a}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathfrak{b}}^{1:t\intercal})$, i.e., the probability of a full instantiation $\vec{x}^k, \vec{a}^k$ at an arbitrary timestep k, given (partial) *evidence* $\vec{z}^{0:t}, \vec{\mathfrak{b}}^{1:t}$ until an arbitrary timestep t. The answer to a query for a probability of an instantiation *contradicting observations*, i.e., an observed state variable $X_i^k \in \vec{Z}^k$ is contradictorily defined by a $z_j^k \in \vec{z}^k$ and $x_i^k \in \vec{x}^k$ or an observed activator random variable $A_{ij}^k \in \vec{\mathcal{B}}^t$ is contradictorily defined by a $\mathfrak{b}_{cv}^k \in \vec{\mathfrak{b}}^k$ and $a_{ij}^k \in \vec{a}^k$, is defined to be of probability zero, e.g., $P(+c, +d, +e | \neg d) = 0$. A query $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathfrak{b}}^{1:t\intercal})$ for an uninstantiated set of random variables $\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}$ is a *query for a distribution* and is answerable by queries for all possible instantiations of $\vec{X}^k, \vec{\mathcal{A}}^k$. ▲

As discussed earlier, inference must be based solely on regular instantiations. If observations are supposed to enforce regularity, we talk about regular or acyclic observations.

Definition 6 (Regular and acyclic observations)**.** An *observation is regular/acyclic*, if every instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ for which $P(\vec{x}^{0:t\intercal}, \vec{a}^{1:t\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathfrak{b}}^{1:t\intercal}) > 0$ holds is regular/acyclic. ▲

Note that this restriction is never enforced, i.e., it is possible to perform inference in non-regular ADBNs if observations are not regular. In such a particular situation, obtained results are not well-defined, but, still, results are obtained. As discussed in Section 8 and by Motzek and Möller (2015a) this circumstance is highly beneficial, as not only acyclicity is a regularity constraint, i.e., obtained results might show up to be well-defined after further research.

Notation 5 (Summation over uninstantiated random variables)**.** *Let R be an uninstantiated, i.e., an unobserved and unqueried, random variable. Then, let $\sum_R P(R)$ denote a* summation over all possible instantiations r *of R, $r \in \mathrm{dom}(R)$, e.g., for a binary R: $\sum_R P(R) = P(+r) + P(\neg r)$. Let $\vec{R}$ denote a column vector of uninstantiated random variables $R_i \in \vec{R}, 0 \leq$*

$i \leq n$. *Then let* $\sum_{\vec{R}} P(R_0, R_1, \dots R_n)$ *denote a* nested summation over all uninstantiated random variables $\sum_{R_i \in \vec{R}}$, *i.e., a summation over all possible instantiation combinations of all random variables in* $\vec{R}$. *For example, for binary* $\vec{R} = \langle R_0, R_1 \rangle^\intercal$*:* $\sum_{\vec{R}} P(R_0, R_1) = P(+r_0, +r_1) + P(+r_0, \neg r_1) + P(\neg r_0, +r_1) + P(\neg r_0, \neg r_1)$.

Proposition 5 (Answering queries about partial subset of instantiations)**.** All queries about partial subsets of instantiations $\vec{x}_q^k$, $\vec{a}_q^k$ of random variables $\vec{X}_q^k \in \vec{X}^k$ and $\vec{\mathcal{A}}_q^k \in \vec{\mathcal{A}}^k$ are answerable by marginalization from $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ as

$$P(\vec{x}_q^{k\intercal}, \vec{a}_q^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \sum_{\vec{X}_u^k} \sum_{\vec{\mathcal{A}}_u^k} P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \ ,$$

where $\vec{\mathcal{A}}_u^k$ is the vector of unbound ("unqueried") activator variables $\vec{\mathcal{A}}_u^k = \vec{\mathcal{A}}^k \backslash \vec{\mathcal{A}}_q^k$ and $\vec{X}_u^k$ is the vector of unqueried state variables $\vec{X}_q^k = \vec{X}^k \backslash \vec{X}_q^k$. Given the distribution $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$, marginalization is exponential in the dimension of unqueried variables from timestep k and in the largest domain $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ of unqueried random variables $X_+ \in \vec{X}_u^k$, $A_{++} \in \vec{\mathcal{A}}_u^k$, i.e., in $\mathcal{O}(|\mathrm{dom}(X_+)|^{|\vec{X}_u^k|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}_u^k|})$. ▲

5.1 Filtering Queries and Problems

Queries $P(\vec{x}^{k\intercal}, \vec{a}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ with $k = t$ to a probabilistic knowledge base are commonly known as filtering queries. Finding answers to queries of the type $P(\vec{x}^{t\intercal}, \vec{a}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ poses two different filtering problems depending on the availability of previous answers to similar queries.

Given historical information about a probabilistic process, filtering queries for multiple timeslices from a broad timerange considering long periods of evidences need to be answered. To answer these queries efficiently, the offline filtering problem needs to be solved.

Definition 7 (Offline filtering problem)**.** Given a probabilistic knowledge base $(B_0, B_\rightarrow)$, the *offline filtering problem* is the task of determining the conditional probability of random variables at all timeslices $0 \leq j \leq t$ given all obtained evidence $\vec{z}^{0:t}, \vec{b}^{1:t}$ so far. This is, to obtain

$$P(\vec{X}^{j\intercal}, \vec{\mathcal{A}}^{j\intercal} | \vec{z}^{0:j\intercal}, \vec{b}^{1:j\intercal}) \ , \forall j : 0 \leq j \leq t \ .$$

We denote a parametrized offline filtering problem as $\texttt{OffFP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$, where t represents the (unary coded) total number of passed timeslices so far. ▲

The offline filtering problem determines the complete conditional joint probability distribution $P(\vec{X}^{j\intercal}, \vec{\mathcal{A}}^{j\intercal} | \vec{z}^{0:j\intercal}, \vec{b}^{1:j\intercal})$ at every timestep j, called a filtering distribution, such that every filtering query for a timeslice j can be answered directly. Further, an answer to a filtering query about partial instantiations can be derived by marginalization, without determining the complete filtering distribution again.

If knowledge about a distribution's evolution over time is required incrementally from $t-1$ to t for every newly obtained evidence, i.e., queries need to be answered temporally consecutively only for the current timeslice t, one has to solve the online filtering problem.

Definition 8 (Online filtering problem)**.** Given a probabilistic knowledge base $(B_0, B_\rightarrow)$ and a stored solution to the online filtering problem at $t-1$, e.g., $P(\vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal} | \vec{z}^{0:t-1\intercal}, \vec{b}^{1:t-1\intercal})$, the *online filtering problem* is the task of determining the conditional probability distribution of random variables at time t given newly obtained evidence $\vec{z}^t, \vec{b}^t$. This is, to obtain

$$P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \ ,$$

conditioned on the stored solution to the online filtering problem at $t-1$. We denote a parametrized online filtering problem as $\texttt{OnlFP}(B_0, B_\rightarrow, \vec{z}^{t-1:t}, \vec{b}^{t-1:t}, t)$, which includes a solution to $\texttt{OnlFP}(B_0, B_\rightarrow, \vec{z}^{t-2:t-1}, \vec{b}^{t-2:t-1}, t-1)$. ▲

The following example demonstrates how a filtering query can be used to gain knowledge about potential infringements in the running example.

Example 7 (Filtering)**.** *With Theorem 1 one can actually model cyclic dependencies as desired in Example 4 and build an ADBN for our example as shown in Figure 3 (black). We assume a noisy-or combination for every state X^t and an individual probability of influence of* 0.8*.*

Say, Don and Earl did pass an initial checkup, but Claire did not. At $t = 1$, one observes a document transfer from Claire to Don, one is unsure about one from Don to Earl, i.e., M_{DE} is uninstantiated, and one can neglect all other transfers. As Claire is known to be tainted, one expects her to influence Earl through *Don, analyzable in a filtering query $P(E^1 | \vec{z}^{0:1\intercal}, \vec{b}^{1\intercal})$, with $\vec{z}^{0:1} = (+c^0, \neg d^0, \neg e^0)^\intercal$, and $\vec{b}^1 = (+m^1_{CD}, \neg m^1_{DC}, \neg m^1_{ED}, \neg m^1_{CE}, \neg m^1_{EC})^\intercal$. As $\vec{b}^1$ is regular, one obtains $P(E^1 | \vec{z}^{0:1\intercal}, \vec{b}^{1\intercal}) = \langle 0.8 \cdot 0.8 \cdot 0.5 = 0.32, 0.68 \rangle$ using a cyclic ADBN, i.e., a cyclic ADBN considers that Earl is possibly influenced by Claire through Don and there exists a probability of* 0.32 *that Earl is now tainted. A diagonal DBN cannot anticipate the indirect influence, because t_1-Earl is influenced by a t_0-Don that has not received a document from Claire. Therefore, a diagonal DBN pretends that Earl is* not *tainted, i.e., $P(E^1 | \vec{z}^{0:1\intercal}, \vec{b}^{1\intercal}) = \langle 0, 1 \rangle$ and there is absolutely no possibility that Earl is tainted.*

To obtain a correct answer in a diagonal DBN for this situation, one needs observations at a finer time scale where indirect influences are not evident during one timeslice. For example, one first has to observe m^1_{CD}, considered in an evaluation of $P(E^1 | \vec{z}^{0:1\intercal}, \vec{b}^{1\intercal})$, and then insert a "correcting" "time"-slice $t = 1.1$, where possibilities of M^1_{DE} are considered during the evaluation of $P(E^{1.1} | \vec{z}^{0:1.1\intercal}, \vec{b}^{1:1.1\intercal})$. To achieve the result of one ADBN evaluation, one needs $n-1$ "diagonal"-evaluations. Somehow, a reasoning framework would need to map queries to serialized observations in a diagonal DBN. Introducing obscure "correcting" timeslices in this manner degrades a BN to a reasoning tool that exists solely for the purpose of solving one inference problem.

Example 7 demonstrates the importance of considering indirect causes, as otherwise highly unexpected results are obtained. Moreover, it demonstrates that by the use of cyclic ADBNs, indirect causes are correctly represented and anticipated, in contrast to approaches using diagonal DBNs. The correct anticipation of indirect influences is achieved by cyclic dependencies in an ADBN, which, as stated by the following theorems, do not cause any overhead in solving associated filtering problems.

Theorem 2 (Exact solution to the offline filtering problem)**.** Given an offline filtering problem $\texttt{OffFP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\mathcal{b}}^{1:t}, t)$, finding an exact solution is linear in t. Finding an exact solution is exponential in the maximal dimension of unobserved variables $\vec{\zeta}^*$, $\vec{\beta}^*$ in a timestep $0 < * < t$, and in the largest domain $\mathrm{dom}(\zeta_+)$, $\mathrm{dom}(\beta_+)$ of all random variables $\zeta_+ \in \vec{\zeta}^{0:t}$, $\beta_+ \in \vec{\beta}^{1:t}$. Finding an exact solution to the offline filtering problem is exponential in the dimension of number of random variables $|\vec{X}^t|$, $|\vec{\mathcal{A}}^t|$ and a respective maximal domain size $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ of all random variables $X_+ \in \vec{X}^t$, $A_{++} \in \vec{\mathcal{A}}^t$. ▲

Theorem 2 is proven by showing that an algorithm exists for finding an exact solution to $\texttt{OffFP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\mathcal{b}}^{1:t}, t)$ in time-complexity $\mathcal{O}(t \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\mathrm{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\mathrm{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and with space-complexity $\mathcal{O}(t \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ for storing $P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal})$ for every timeslice t. The proof is combined with the proof for the following theorem.

Theorem 3 (Exact solution to the online filtering problem)**.** Given an online filtering problem $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{t-1:t}, \vec{\mathcal{b}}^{t-1:t}, t)$, finding an exact solution is constant in t. Finding an exact solution is exponential in the dimension of unobserved variables from timestep $t-1$ and in the largest domain $\mathrm{dom}(\zeta_+)$, $\mathrm{dom}(\beta_+)$ of random variables $\zeta_+ \in \vec{\zeta}^{t-1}$, $\beta_+ \in \vec{\beta}^{t-1}$. Finding an exact solution to the online filtering problem is exponential in the dimension of number of random variables $|\vec{X}^t|$, $|\vec{\mathcal{A}}^t|$ and a respective maximal domain size $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ of all random variables $X_+ \in \vec{X}^t$, $A_{++} \in \vec{\mathcal{A}}^t$. ▲

These theorems discuss the computational complexity of filtering problems in general ADBNs, and, most importantly, state that filtering in ADBNs still remains constant over time, as one is used to for classical DBN filtering problems. Both theorems are proven in Appendix B whose major result is a commonly known recursive definition of a filtering equation, given in the following corollary, for solving filtering problems.

Corollary 1 (Filtering equation)**.** Using a recursive definition, an exact solution to a filtering problem for a consecutive timestep based on one of the current timestep in a dense intra-timeslice ADBN is given by

$$P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal}) = \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} P(\vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal} | \vec{z}^{0:t-1\intercal}, \vec{\mathcal{b}}^{1:t-1\intercal}) \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \setminus X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \; . \quad ▲ \quad (6)$$

This corollary is an excerpt of the proof for Theorems 3 and 2 in Appendix B, showing that filtering has constant complexity w.r.t. time in dense intra-timeslice ADBNs, i.e., the most general form of cyclic ADBNs. Moreover, it shows that *no novel calculus* (in significant contrast to Milch et al., 2005; Bilmes, 2000; Geiger & Heckerman, 1996) is required to perform exact inference in dense intra-timeslice ADBNs, despite the fact that ADBNs might be based on cyclic graphs. Furthermore, note that at no timestep an effective structure is known in advance and indeed remains unknown, and, still, the general filtering equation does not even require a postponed analysis of such a structure.

If one would be constrained to achieve similar results in an acyclic, diagonal (A)DBN, a serialized observation set with spurious intermediate timeslices had to be generated. Filtering in such a generated diagonal network would then be n-times slower. Additionally, a generated order needs to be stored for answering queries. We refrain from a detailed analysis on how similar results could be achieved in a diagonal (A)DBN by artificially serializing observations; a diagonal (A)DBN simply represents the wrong model and cyclic ADBNs are immediately required to provide a local and causal parametrization of such models.

5.2 Prediction

is used to propagate a possible evolution into the future. Essentially, prediction is a filtering query with an empty observation set. If observations are supposed to enforce regularity, plain prediction is not possible in our formalism, as a minimal set of observations is needed to ensure regularity. Nevertheless, one could use a *most likely acyclic observation* for prediction. Finding a most likely acyclic observation is a special case of a most likely explanation problem. Later in Section 7, we introduce a solution based on our formalism with full support for prediction without requiring specific observations to enforce regularity.

5.3 Smoothing Queries and Problems

Queries $P(\vec{x}^{k\intercal}, \vec{a}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ with $k < t$ to a probabilistic knowledge base are commonly known as smoothing queries and are used to obtain knowledge in retrospect. Finding answers to queries $P(\vec{x}^{k\intercal}, \vec{a}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$, $k < t$ defines two smoothing problems: complete and fixed-lag smoothing.

Similar to the offline filtering problem, the complete smoothing problem needs to be solved when answering multiple smoothing queries for different timeslices from broad timerange of evidences investigated in hindsight.

Definition 9 (Complete smoothing problem)**.** Given a probabilistic knowledge base $(B_0, B_\rightarrow)$, the *complete smoothing problem* is the task of determining the conditional probability of random variables at all times $0 \leq j < t$, considering evidence $\vec{z}^{0:t}, \vec{b}^{1:t}$ until time t. This is, to obtain

$$P(\vec{X}^{j\intercal}, \vec{\mathcal{A}}^{j\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \ , \ \forall j : 0 \leq j < t \ .$$

We denote a parametrized complete smoothing problem as $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$. ▲

Solutions to the complete smoothing problem are used to investigate a distribution's evolution over time in retrospect, but by considering evidence upto a later timepoint.

Similar to the online filtering problem, the fixed-lag smoothing problem needs to be solved, if smoothing queries need to be answered temporally consecutively for every newly obtained evidence.

Definition 10 (Fixed-lag smoothing problem)**.** Given a probabilistic knowledge base $(B_0, B_\rightarrow)$ and a solution to the respective fixed-lag smoothing problem at $t-1$, the *fixed-lag smoothing problem* at time t is the task of determining the conditional probability distribution of random variables at a time $k = t - \Delta$, considering evidence $\vec{z}^{0:t}, \vec{b}^{1:t}$ until time t. This is, to obtain

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \ .$$

We denote a parametrized fixed-lag smoothing problem as $\texttt{FLagSP}(B_0, B_{\to}, \vec{z}^{t-\Delta-1:t}, \vec{\mathcal{B}}^{t-\Delta-1:t}, \Delta, t)$, which includes a solution to $\texttt{FLagSP}(B_0, B_{\to}, \vec{z}^{t-\Delta-2:t-1}, \vec{\mathcal{B}}^{t-\Delta-2:t-1}, \Delta, t-1)$. ▲

Answers to the fixed-lag smoothing problem are used to track a distribution over time. Fixed-lag smoothing "lags" Δ timeslices behind real time, but a trajectory is smoothed out by a look ahead in time (cf. Russell & Norvig, 2010, p. 571).

The following example explains the application of a smoothing query to the running example and accentuates the need for cyclic ADBNs in favor of diagonal DBNs. Continuing Example 7 the example demonstrates that smoothing handles explaining away over multiple timesteps and respects indirect causes.

Example 8 (Explaining away)**.** *We assume that only Don underwent a successful compliance check at time $t = 0$, i.e., $\vec{z}^0 = (\neg d^0)$. For $t = 1$ the same document transfers as in Example 7 were detected, and for $t = 2$, a Sunday, all message transfers negligible, i.e., $\vec{\beta}^2 = \emptyset$. On that Sunday irregularities in Earl's documents were found, i.e., $\vec{z}^2 = (+e^2)$.*

If one performs a smoothing query for Claire's initial belief state ($t = 0$) without considering evidence from $t = 2$, an answer is equivalent to the prior belief of Clare, i.e., $P(C^0) = P(C^0|\vec{z}^{0:1\intercal}, \vec{\mathcal{B}}^{1\intercal}) = \langle 0.5, 0.5\rangle$, as one has not gained any new information with the evidence from $t = 1$. However, with observations from $t = 2$, one needs to consider an indirect influence by Claire onto Earl and the belief in Claire being tainted rises to $P(C^0|\vec{z}^{0:2\intercal}, \vec{\mathcal{B}}^{1:2\intercal}) \approx \langle 0.532, 0.468\rangle$, because the observation $+e^2$ tells one indirectly something about $+c^0$.

The slow increase from $P(+c^0) = 0.5$ to $P(+c^0|\vec{z}^{0:2\intercal}, \vec{\mathcal{B}}^{1:2\intercal}) = 0.532$ is due to the high prior belief in Earl manipulating documents of $P(+e^0) = 0.7$ and it is more likely that Earl has been manipulating documents ever since. If, say, Earl can be relieved from initial incrimination, i.e., $\neg e^0$, the only explanation for this situation is an indirect cause of Claire being tainted, and that Claire has influenced Earl through Don at time $t = 2$, which is correctly handled as $P(+c^1, +m^1_{DE}|\vec{z}^{0:2\intercal}, \vec{\mathcal{B}}^{1:2\intercal}) = 1$. One can further update an initial prior belief using a smoothing query and find that $P(+d^0) = P(+e^0) = 0$ but $P(+c^0) = 1$. This means that the only explanation for the observations made is that Claire has been corrupt *from the beginning ($+c^0$) and that Don has actually sent a message to Earl ($+m^1_{DE}$), i.e., the possibility of $\neg m^1_{DE}$ is explained away due to the observations made at times $t = 0 \dots 2$.*

In a diagonal (A)DBN the last example is inexplicable, as indirect influences of t_1 (causally) are first anticipated one step later at t_2 (for $n = 3$). The reason for the inexplicability is confusing because it is not causal: at t_2, the time of incriminating evidence for Earl, one knows that Earl is only influenced by himself, i.e., only t_1-Earl can be the source of his taintedness. At t_1, Earl only receives a document from integrous t_0-Don (observation). This is where the problem lies: t_0-Claire should have influenced t_0-Don by now, but t_0-Claire influences t_1-Don with her message $+m^1_{CD}$. This means that Earl cannot become tainted and the observation $+e^2$ is inexplicable. Mathematically one obtains $P(+e^0|\vec{z}^{0:2\intercal}, \vec{\mathcal{B}}^{1:2\intercal}) = 0$ in a diagonal DBN, as

$$P(+e^0|\vec{z}^{0:2\intercal}, \vec{\mathcal{B}}^{1:2\intercal}) = \alpha \cdot \sum_{C^0} P(C^0, \neg d^0, \neg e^0) \cdot \sum_{C^1} P(C^1|\dots, C^0) \cdot \sum_{D^1} P(D^1|\dots, C^0, \dots)$$

$$\cdot \sum_{M^1_{DE}} P(+m^1_{CD}, \neg m^1_{DC}, \neg m^1_{ED}, \neg m^1_{CE}, \neg m^1_{EC}, M^1_{DE})$$
$$\cdot \Big(P(\underline{+e^1} | \underline{\neg m^1_{CE}}, M^1_{DE}, C^0, \underline{\neg d^0}, \underline{\neg e^0}) \cdot P(\neg e^2 | \neg m^2_{CE}, \neg m^2_{DE}, C^1, D^1, +e^1)$$
$$+ P(\neg e^1 | \neg m^1_{CE}, M^1_{DE}, C^0, \neg d^0, \neg e^0) \cdot P(\underline{\neg e^2} | \underline{\neg m^2_{CE}}, \underline{\neg m^2_{DE}}, C^1, D^1, \underline{\neg e^1}) \Big)$$
$$\cdot P(\neg m^2_{**}) = 0$$

where both alternatives of E^1 are impossible under the given observations, and CPD entries $P(+e^2 | \underline{\neg m^2_{*E}}, C^1, D^1, \underline{\neg e^1})$ and $P(+e^1 | M^1_{DE}, \underline{\neg m^1_{CE}}, C^0, \underline{\neg d^0}, \underline{\neg e^0})$ are uniquely identified to be 0 by the underlined dependencies. By definition, one obtains $P(\neg e^2 | \ldots, +e^2, \ldots) = 0$, and, thus, obtained results from a diagonal DBN stand in conflict with the probability axioms of Kolmogorov (compare Proposition 2 under which $\vec{\mathcal{B}}^{1:2}$ is not indirect-free). As Example 8 shows, the observation of Example 8 is regular and an intra-timeslice ADBN fully respects indirect influences while remaining a first-class representation.

Example 8 repeatedly shows that diagonal (A)DBNs are too restricted in their expressivity to represent the taintedness domain correctly, and it shows that cyclic ADBNs arise naturally and are required to provide locally and directly understandable CPDs. The following theorems state that these novelly allowed cycles do not cause any overhead in solving classically known smoothing problems in dense intra-timeslice ADBNs.

Theorem 4 (Exact solution to the complete smoothing problem)**.** Given a complete smoothing problem $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathcal{B}}^{1:t}, t)$, finding an exact solution is linear or quadratic in t. Finding an exact solution is exponential in the maximal dimension of unobserved variables $\vec{\zeta}^*$, $\vec{\beta}^*$ in a timestep $0 < * \leq t$, and in the largest domain $\mathrm{dom}(\zeta_+)$, $\mathrm{dom}(\beta_+)$ of all random variables $\zeta_+ \in \vec{\zeta}^{0:t}$, $\beta_+ \in \vec{\beta}^{1:t}$. Finding an exact solution is exponential in the dimension of number of random variables $|\vec{X}^t|$, $|\vec{\mathcal{A}}^t|$ and a respective maximal domain size $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ of all random variables $X_+ \in \vec{X}^t$, $A_{++} \in \vec{\mathcal{A}}^t$. ▲

Theorem 4 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathcal{B}}^{1:t}, t)$ in time-complexity $\mathcal{O}(t^2 \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\mathrm{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\mathrm{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and with $\mathcal{O}(t \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ space-complexity for storing all smoothing distributions. The proof is combined with the proof for the following theorem.

Theorem 5 (Exact solution to the fixed-lag smoothing problem)**.** Given a fixed-lag smoothing problem $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{t-\Delta-1:t}, \vec{\mathcal{B}}^{t-\Delta-1:t}, \Delta, t)$, finding an exact solution is constant in t. Let $k = t - \Delta$ for brevity. Finding an exact solution is exponential in the maximal dimension of unobserved variables $\vec{\zeta}^*$, $\vec{\beta}^*$ in a timestep $k - 1 < * \leq t$, and in the largest domain $\mathrm{dom}(\zeta_+)$, $\mathrm{dom}(\beta_+)$ of all random variables $\zeta_+ \in \vec{\zeta}^{k-1:t}$, $\beta_+ \in \vec{\beta}^{k-1:t}$. Finding an exact solution is exponential in the dimension of number of random variables $|\vec{X}^t|$, $|\vec{\mathcal{A}}^t|$ and a respective maximal domain size $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ of all random variables $X_+ \in \vec{X}^t$, $A_{++} \in \vec{\mathcal{A}}^t$. ▲

These theorems discuss the computational fixed-parameter complexity of commonly known smoothing problems in dense intra-timeslice ADBNs. Most importantly, Theorem 4 states that solving the complete smoothing problem has linear complexity over the number of observed timeslices, as one expects in classical DBNs, and Theorem 5 states that solving the

fixed-lag smoothing problem has constant complexity over the number of observed timeslices, as one expects in classical DBNs as well. Both theorems are proven in Appendix C, whose major result is the following corollary, which shows that the classically known forward-backward-algorithm still remains applicable.

Corollary 2 (Smoothing equation)**.** The general smoothing equation to obtain a smoothing distribution at time k, given evidence until time t is defined by

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal}) = P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{\mathcal{b}}^{1:k\intercal}) \cdot P(\vec{z}^{k+1:t\intercal}, \vec{\mathcal{b}}^{k+1:t\intercal} | \vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}) \ , \tag{7}$$

where the first term represents a filtering problem (compare Corollary 1), and the second term represents a so-called backward-message. Using a recursive definition of the backward-message one obtains

$$\begin{aligned} P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal}) = \alpha \cdot P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{\mathcal{b}}^{1:k\intercal}) \\ \cdot \sum_{\vec{\zeta}^{k+1}} \sum_{\vec{\beta}^{k+1}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1\intercal} \setminus X_i^{k+1}, \vec{A}_i^{k+1\intercal}, X_i^k) \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \\ \cdot P(\vec{z}^{k+2:t\intercal}, \vec{\mathcal{b}}^{k+2:t\intercal} | \vec{X}^{k+1\intercal}, \vec{\mathcal{A}}^{k+1\intercal}) \ . \quad \blacktriangle \end{aligned} \tag{8}$$

This corollary is an excerpt of Proof C and shows that smoothing problems are solvable using a classical calculus without introducing external frameworks or new mathematical operators. Moreover, using this equation one obtains the commonly known forward-backward-algorithm: For every k, all $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{\mathcal{b}}^{1:k\intercal})$ are obtained by solving an offline filtering problem, which is linear in t. By descendingly solving $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal})$ from $k = t-1 \dots 0$, one iteratively obtains $P(\vec{z}^{k+1:t\intercal}, \vec{\mathcal{b}}^{k+1:t\intercal} | \vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal})$ in constant time, and thus solves each $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal})$ in constant time.

5.4 Experimental Evaluation

Sections 5.1 and 5.3 have shown that no novel calculus must be invented to perform inference in dense intra-timeslice ADBNs, even with cyclic graphs, and, moreover, the sections have demonstrated that no computational overhead is introduced to perform inference. We provide substantiating empirical evidence for the latter results by showing that solving multiple on- and offline filtering- and smoothing-problems remains tractable even over large periods of time.[2] To do so, we perform multiple experiments, where in every experiment, state variables in $\vec{X}^t, t > 0$ are assigned a randomly generated individual CPD following Definition 2. Further, in every experiment random priors are assigned to random variables $\vec{X}^0$ and $\vec{\mathcal{A}}^t$. Every randomly generated probability is taken from the range $[0.1, 0.9]$ to avoid impossible observations. For every timestep $t > 0$ we generate random observations $\vec{\mathcal{b}}^t$ conforming with Theorem 1. We refrain from observing state variables $\vec{X}^t$, i.e., $\vec{z}^t = \emptyset$, in order to achieve worst-case time complexity.

In every experiment, we consecutively solve the online filtering problem $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{i-1:i}, \vec{\mathcal{b}}^{i-1:i}, i)$ at every timestep i using the algorithm derived in Proof B, and denote the computation time per timeslice i in Figure 4. Further, we solve the complete smoothing

2. All experiments are reproducible, for which we supply an experimental framework implementing inference in dense ADBNs in C available at: http://adbn.motzek.org

problem $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:i}, \vec{b}^{1:i}, i)$ at every timestep i, using the algorithm derived in Proof C using stored filtering results (Figure 6) and recalculated filtering results (Figure 5) and denote the computation time per timeslice i.

As $\vec{z}^t = \emptyset$, one does not acquire any new knowledge about state variables using smoothing, i.e., $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{b}^{1:k\intercal})$ is equal to $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$. A Kullback-Leibler-Divergence of both solutions was measured to be zero (in the range of double precision), which verifies our implementation in that sense.

Experiments were repeated 139 times for $n = 4$, i.e., an ADBN consisting of 16 random variables $\vec{X}^t, \vec{\mathcal{A}}^t$ per timeslice for a timerange of 40. All experimental results validate expected fixed-parameter tractability of filtering and smoothing problems in ADBNs. Nevertheless, experiments show that for models beyond $n = 5$, i.e., beyond 25 random variables, approximate inference techniques are needed, for which the following section provides an introduction and a demonstration.

6. Approximate Inference Techniques in ADBNs

Finding exact solutions to common problems is only tractable in small toy domains, i.e., in domains with very few random variables. Approximate inference techniques have shown to be a valuable alternative for Bayesian networks and can be divided into two categories: (i) approximations to the derivation of exact calculations, such as loopy belief propagation or variational methods, and (ii) stochastic sampling methods on which we focus in this section. Introductions to sampling methods for general graphical probabilistic models from a stochastical perspective are provided by Arulampalam, Maskell, Gordon, and Clapp (2002), Murphy (2012, pp. 823–831), and Doucet and Johansen (2009).

Naive adaptations of approximate inference techniques for BNs towards DBNs, however, show to be suboptimal, as approximation errors accumulate over time, i.e., only for a few consecutive timeslices accurate results are obtained. Fortunately, modifications to these approaches allow for approximate inference in DBNs even for long periods of inference time under a bounded and constant approximation error. Such a property of an approximation algorithm, namely, providing a bounded and constant error over time, must be preserved for any novel DPGM such as ADBNs. Therefore, we show in this section that commonly known "particle filters", aka sequential important resampling (SIR) techniques, remain applicable in ADBNs without any overhead, and provide approximations with constant error over time. On top of that, an interesting challenge remains: approximate inference techniques for DBNs are based on the topological ordering of $B_\rightarrow$. However, a topological ordering of a (dense intra-timeslice) ADBN is firstly known in a certain instantiation. Therefore, an approximate inference technique must rapidly adapt to a specific context at every sampling step.

Remark 3 (Approximate inference nomenclature)**.** *Sampling based approximate inference techniques for DBNs are referred to under various names (cf. Russell & Norvig, 2010, pp. 603ff). In general, every sampling based technique can be classified as a* Monte Carlo *simulation. For DBNs, two major sampling based approximations are known as* sequential importance sampling *(SIS) and* sequential importance resampling *(SIR), both are sometimes referred to as* sequential Monte Carlo *(SMC) techniques (cf. Arulampalam et al., 2002). Furthermore, SIS and SIR are often referred to as* particle filters *(especially by Murphy, 2012, pp. 823–831 and Doucet, de Freitas, Murphy, & Russell, 2000). We adapt the naming of SIS and SIR.*

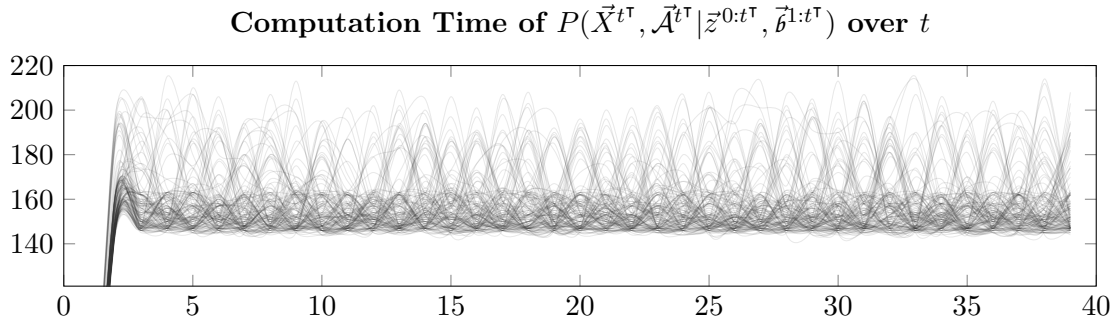

Figure 4: Finding an exact solution to the online filtering problem $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{i-1:i}, \vec{b}^{i-1:i}, i)$ at timestep i (abscissa) is constant (computation time in ms, ordinate) at every timestep i. (139 evaluations superimposed)

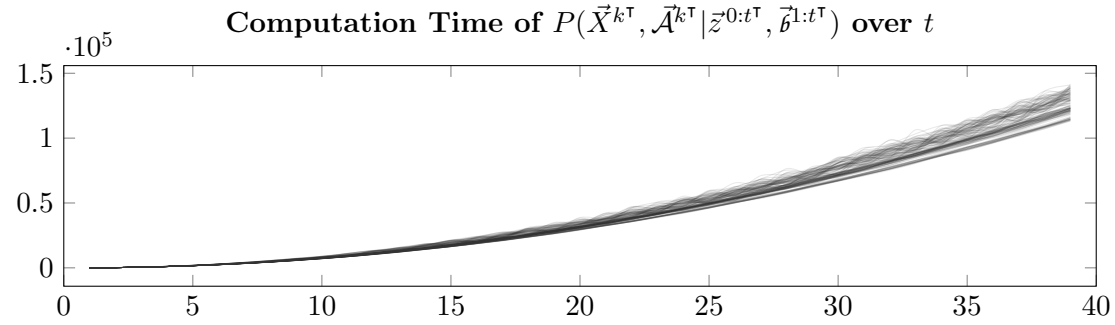

Figure 5: Finding an exact solution to the complete smoothing problem $\texttt{ComplSP}(B_0, B_{\rightarrow}, \vec{z}^{0:i}, \vec{b}^{1:i}, i)$ at timestep i (abscissa), without stored filtering results, scales quadratically (computation time in ms, ordinate) with increasing timesteps, but has constant memory requirements.

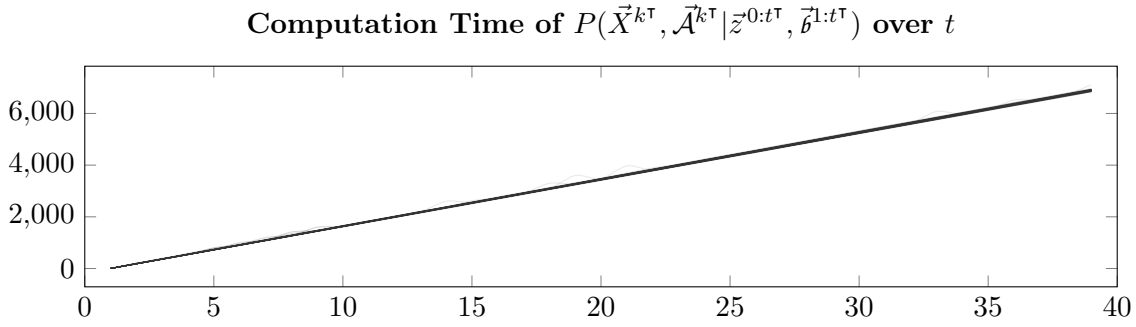

Figure 6: Finding an exact solution to the complete smoothing problem $\texttt{ComplSP}(B_0, B_{\rightarrow}, \vec{z}^{0:i}, \vec{b}^{1:i}, i)$ at timestep i (abscissa), using stored filtering results scales linearily (computation time in ms, ordinate) with increasing timesteps, but has linear memory requirements over time.

The key idea behind sampling approaches is to estimate a probability distribution, e.g., $P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal})$, by a large number of samples, instead of performing exact inference. This means that a modeled stationary process over time is simulated multiple times, where every simulation generates a sample S with a specific outcome.

Definition 11 (Sample)**.** Let $S = \left(\vec{x}^t, \vec{a}^t, w\right)$ denote a *sample of random variables* $\vec{X}^t, \vec{\mathcal{A}}^t$ at time t, i.e., an arbitrary instantiation $\vec{x}^t, \vec{a}^t$ of $\vec{X}^t, \vec{\mathcal{A}}^t$. Each sample is assigned a weight w. A sample S is sometimes called a particle.

Let $\vec{S}^t$ denote the *set of all samples* obtained at time t. Let $n_S = |\vec{S}^t|$ be the number of samples per timeslice. Let w_S denote the assigned *weight* w of a sample S. Let $S^t_{\vec{x},\vec{a}}$ denote a sample with instantiation $\vec{x}^t, \vec{a}^t$ and let $\vec{S}^t_{\vec{x},\vec{a}}$ denote the set of all samples $S_{\vec{x}^t,\vec{a}^t}$ obtained at time t. We use $S^t_{\vec{x},\vec{a}}$ in favor of a notation for continuous state spaces using the delta Dirac mass distribution located at $\vec{x}^t, \vec{a}^t$ (cf. Murphy, 2012, pp. 823–831; Doucet & Johansen, 2009). ▲

The key challenge behind approximate inference techniques is to efficiently and correctly generate these samples for a given (dynamic) probabilistic graphical model and a set of observations. In the following section we discuss two approaches, namely, *sequential importance sampling* (SIS) and *sequential importance resampling* (SIR).

6.1 SIS and SIR in ADBNs

In order to approximate solutions to filtering and smoothing problems, a distribution of obtained samples must correspond to the distribution they shall approximate. For classic Bayesian networks multiple techniques exist to generate stochastically correct samples, namely "prior sampling," "rejection sampling," "likelihood weighting," and "Gibbs sampling" (cf., Russell & Norvig, 2010, pp. 530–535). SIS and SIR represent an adaptation of likelihood weighting to DBNs. In SIS and SIR samples are sequentially updated at each timestep and are weighted according to their importance, i.e., their conformity with evidence.

Informally, one can describe SIS for a generic DBN $(B_0, B_\rightarrow)$ with state variables $\vec{X}^t$ as follows: To update a sample S^{t-1}, i.e., an arbitrary instantiation of random variables at $t-1$, to an updated sample S^t, one follows the stochastic model of the DBN: Following the topological ordering of $B_\rightarrow$ every random variable X^t_i is (a) observed or (b) unobserved. In case (a) the instantiation of X^t_i in the updated sample S^t is fixed to the observed value $x^t_i \in \vec{z}^t$ and weighted according to conformity of the current evolution $S^{t-1} \rightarrow S^t$ with the observed evidence $P(x^t_i | S^{t-1}, S^t)$. In case (b) the instantiation of X^t_i is sampled, i.e., is randomly instantiated, corresponding to its probability distribution conditioned on the current evolution, i.e., sampled according to $P(X^t_i | S^{t-1}, S^t)$. S^{t-1} is a full instantiation of all random variables at $t-1$, and S^t is generated sequentially according to a topological ordering, therefore $P(X^t_i | S^{t-1}, S^t)$ is a locally defined CPD.

Adapting SIS to a dense intra-timeslice ADBN with variables $\vec{X}^{0:t}$ and $A^{1:t}$ gives an update procedure as Algorithm 1 that is used in sequential importance (re)sampling techniques as shown in Algorithm 2. This update procedure requires an effective topological ordering in a timestep t based on the observed activator random variables in $\vec{\mathcal{b}}^t$.

Notation 6 (Sampling operator $\leftarrow\!\!\boxtimes$). *$X \leftarrow\!\!\boxtimes\, P(X)$ represents a* sampling operation. *The sampling operation instantiates a random variable X randomly to one of its possible values $x \in \mathrm{dom}(X)$ according to a given random distribution P of X for all $x \in \mathrm{dom}(X)$.*

Algorithm 1 Evolving a Sample

procedure EVOLVE(from S^{t-1} to S^t, given $\vec{z}^t$, $\vec{\mathcal{b}}^t$)
  for all $A_{ij}^t \in \vec{\mathcal{A}}^t$ **do** ▷ skip at t=0
    if $a_{ij}^t \in \vec{\mathcal{b}}^t$ **then** ▷ is observed
      $w \leftarrow w \cdot P(a_{ij}^t)$
      set $A_{ij}^t \leftarrow a_{ij}$ in S^t
    else
      set $A_{ij}^t \leftarrow\!\!\boxtimes\, P(A_{ij}^t)$ in S^t
  for all $X_i^t \in \vec{X}^t$ following topological ordering induced by $\vec{\mathcal{b}}^t$ **do**
    if x_i^t is observed in $\vec{z}^t$ **then**
      $w \leftarrow w \cdot P(x_i^t | \vec{x}^t, \vec{a}_i^t, x_i^{t-1})$
      set $X_i^t \leftarrow x_i$ in S^t
    else
      set $X_i^t \leftarrow\!\!\boxtimes\, P(X_i^t | \vec{x}^t, \vec{a}_i^t, x_i^{t-1})$ in S^t

The analysis of an effective topology is necessary to sample from the correct distribution and to correctly weight a sample. Although Algorithm 1 uses local CPDs conditioned on instantiations of $\vec{X}^t$ that are not yet updated, all uninstantiated dependencies are deactive due to the induced topological ordering of $\vec{\mathcal{b}}^t$.

Algorithm 2 Sequential Importance (Re)Sampling (SIS/SIR)

begin
  init $\vec{S}$ with n_S $S \leftarrow (\emptyset, \emptyset, 1)$ samples
  set $t \leftarrow 0$
  loop
    given $\vec{z}^t$, $\vec{\mathcal{b}}^t$ ▷ t=0: $\vec{\mathcal{b}}^t = \emptyset$
    for all samples $S \in \vec{S}$ **do**
      $S \leftarrow$EVOLVE$(S, \vec{z}^t, \vec{\mathcal{b}}^t)$
    for all samples $S \in \vec{S}$ **do**
      $w_S \leftarrow \frac{w_S}{\sum_{S' \in \vec{S}} w_{S'}}$
    $\vec{S} \leftarrow\!\!\boxtimes\, P(\vec{S})$ ▷ Only for SIR.
    next t
  return $\vec{S}$
end

Algorithms 1 and 2 represent a straightforward adaptation of classically known SIS and SIR algorithms towards dense intra-timeslice ADBNs. In the following we show that occurring cycles in an ADBN do not prevent these algorithms from obtaining accurate results to commonly known filtering problems. Moreover, we show that still a bounded error over

time is achieved. In order to do so, Algorithm 2 must provide an exact solution to the online and offline filtering problem, as stated by the following theorem.

Theorem 6 (Exact sampling based solutions to filtering problems)**.** For $n_S \to \infty$ and infinite numerical precision, samples $\vec{S}^t$ generated by Algorithm 2 (SIS) represent the filtering distribution $P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ with

$$P(\vec{x}^{t\intercal}, \vec{a}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \frac{\sum_{S \in \vec{S}_{\vec{x}^t, \vec{a}^t}} w_S}{\sum_{S \in \vec{S}} w_S} \ . \tag{9}$$

Thus, Algorithm 2 solves the offline filtering problem (Definition 7) in linear time-complexity over time, scales linearly with the number of samples n_S and linearly with the number of random variables $|\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|$, i.e., it solves $\texttt{OffFP}(B_0, B_\to, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$ in $\mathcal{O}(t \cdot n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ space- and time-complexity. Initializing Algorithm 2 with samples $\vec{S}$ from a previous solution $\vec{S}^{t-1}$ from $t-1$ is an exact solution to the online filtering-problem (Definition 8) with constant time-complexity over time $\mathcal{O}(1)$, i.e., it solves $\texttt{OnlFP}(B_0, B_\to, \vec{z}^{t-1:t}, \vec{b}^{t-1:t}, t)$ in $\mathcal{O}(n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ time and space complexity. ▲

This theorem shows that for infinite samples, a classical SIS procedure remains applicable to cyclic ADBNs. Theorem 6 is proven in Appendix D. For finite amounts of samples and finite numerical precision, SIS and SIR deliver approximate solutions for filtering problems. However, for finite amounts of samples, SIS techniques suffer from a "degeneracy" problem (cf., Murphy, 2012, pp. 823–831; Doucet & Johansen, 2009) in DBNs, as well as in ADBNs: Gradually, a significant amount of samples degrades and carries a very low weight, i.e., during sequential updating, sampled instantiations do not fit to evidence at later timepoints. Therefore, a large amount of samples become practically irrelevant for the approximation and only few samples left carry a high weight and are highly important. This means that an approximation error does not remain bounded over time and gradually increases. To overcome this problem a resampling procedure is introduced in SIR, which resamples n_S samples from a current sample distribution (Alg. 2, Line 10) according to their importance-weights w_S—the reason why some people speak of the "survival of the fittest" in SIR techniques. As a resampling technique we use stratified resampling as presented by Hol, Schön, and Gustafsson (2006), which showed to be more accurate and faster than multinomial resampling (cf. Hol et al., 2006; Massey, 2008). The following proposition states that SIR procedures remain applicable for cyclic ADBNs also for finite amounts of samples.

Proposition 6 (Approximate sampling based solutions to the online filtering problem)**.** For finite number of samples n_S, finite numerical precision and non-zero probabilities in every defined CPD, SIR, as in Algorithm 2, provides an approximate solution to the online filtering problem (cf. Theorem 6) with bounded error over time. SIR scales linearly in computation time and memory requirements with the number of samples n_S and linearly with the number of random variables per timestep. Thus SIR solves $\texttt{OnlFP}(B_0, B_\to, \vec{z}^{t-1:t}, \vec{b}^{t-1:t}, t)$ in time-complexity $\mathcal{O}(n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$. Further, an approximate solution to $\texttt{OnlFP}(B_0, B_\to, \vec{z}^{t-1:t}, \vec{b}^{t-1:t}, t)$ obtained by using SIR only requires $\mathcal{O}(n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ storage for all samples from which every filtering distribution can be obtained. Likewise, it provides an approximate solution to the offline filtering problem by solving t online filtering problems, i.e., solves $\texttt{OffFP}(B_0, B_\to, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$ in $\mathcal{O}(t \cdot n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ space- and time-complexity. ▲

Proposition 6 is a derivative of Theorem 6 for the resampling case. The following section provides substantiating evidence for it through an empirical study of the approximation accuracy of a classical SIR applied to a cyclic ADBN, which, in summary, delivers highly accurate approximation results, as expected from SIR.

Moreover, answering queries via a sampling based approximation has significant advantages in computational complexity as well as required storage complexity, as stated by the following proposition.

Proposition 7 (Filtering query answering from samples)**.** Every filtering query $P(\vec{x}_q^{t\intercal}, \vec{a}_q^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$, where $\vec{x}_q^t, \vec{a}_q^t$ are partial instantiations of random variables $\vec{X}_q^t \in \vec{X}^t, \vec{\mathcal{A}}_q^t \in \vec{\mathcal{A}}^t$, are directly answerable from the set of samples $\vec{S}^t$ obtained at time t in $\mathcal{O}(n_S)$, by summation over all samples $\vec{S}_{\vec{x}^t, \vec{a}^t}$ that contain the partial instantiation $\vec{x}_q^t, \vec{a}_q^t$. This reduces the space-complexity of the filtering problem to storing n_S samples for all timeslices, i.e., t timeslices for the offline filtering problem, and 1 for the online filtering problem. Further, this reduces time-complexity of answering a query for any partial instantiation to $\mathcal{O}(n_S)$. ▲

As one expects for classical approximation procedures in DBNs, this proposition shows that a probability distribution has neither to be stored nor to be made explicit to marginalize out answers to queries in cyclic ADBNs. Therefore, plain linear complexity is obtained, which permits the use of ADBNs in largely scaled problems, as empirically evaluated in the following section.

6.2 Experimental Evaluation of Approximate Inference

In the following, we provide substantiating evidence for Proposition 6, by evaluating SIR in terms of approximation accuracy and performance compared to exact inference. To judge the accuracy of an approximated result obtained by SIR, the Hellinger distance between approximated and exactly calculated distribution is used.

Definition 12 (Hellinger distance)**.** Let P_R, P_S, $\operatorname{rank} P_R = \operatorname{rank} P_S = N$ denote two probability distributions over the same N events $p \in P$. The *Hellinger distance* $H(P_R, P_S)$ between both distributions is defined as (cf. Gibbs & Su, 2002)

$$H(P_R, P_S) = \frac{1}{\sqrt{2}} \sqrt{\sum_{p \in P} \left(\sqrt{P_S(p)} - \sqrt{P_R(p)} \right)^2}$$

The Hellinger distance is symmetric, i.e., $H(P_R, P_S) = H(P_S, P_R)$, and is bounded between 0 and 1. ▲

A Hellinger distance of 0 exists between two distributions that assign the same probability to common events (no error). A distance of 1 occurs between two distributions, say, P_S and P_R, that completely contradict each other, i.e., if every event that is deemed impossible in P_S, is supposed to be possible in P_R, and every event possible in P_S is deemed impossible in P_R. Naturally, one expects small differences between approximated and exact results, i.e., low, near 0, Hellinger distances. We refrain from using the more common Kullback-Leibler-Divergence (KLD), as negligible errors can lead to a KLD of ∞. A single event p with $P_R(p) = \varepsilon$, very small $\varepsilon > 0$, but with $P_S(p) = 0$ leads to $KLD(P_R, P_S) = \infty$.

For evaluation we consider the same models as in the previous evaluation of exact inference discussed in Section 5.4. As an evaluation against exact inference requires exact solutions of online filtering problems, we keep $|\vec{X}^t| = 4$ for a detailed evaluation over 250 timesteps (Figure 7, repeated 25 times) and demonstrate it on $|\vec{X}^t| = 5$ for only 25 timesteps (Figure 9, repeated 2 times). For $|\vec{X}^t| = 4$ one state variable X_i^t is observed per timestep and for $|\vec{X}^t| = 5$ two are observed. Observations of state variables lead to the aforementioned degeneracy problem of SIS, which is clearly evident in Figure 7. Further, Figure 7 shows over 250 evaluated timesteps of constant error for SIR, which delivers a highly satisfying approximation accuracy for 10 000 and more samples (Hellinger distance below 0.1). Judging from Figure 7 it seems that the Hellinger distance of an SIR approximation linearly decreases with increasing samples. The constant and low Hellinger distance over long periods of time, evident from Figure 7, shows that SIR indeed delivers an approximate solution to the online and offline filtering problem for a finite number of samples, scaling linearly with the number of random variables (Figure 8), as stated by Proposition 6, which we consider empirically justified.

While an algorithm based on Coroloary 1 finds an exact solution to the online filtering problem for $|\vec{X}^t| = 4$ with one observation almost instantly, it takes more than 12 minutes for $|\vec{X}^t| = 5$ with one observation and around 6 minutes for two observations *per timestep*, as shown in Figure 9 (black). In contrast, an approximate solution with 10 000 000 samples is found by SIR in 16 seconds per timestep in the same experiment. Exact inference on $|\vec{X}^t| = 6$, i.e., 36 random variables in $B_{\rightarrow}$ is impossible, as the probability distribution alone requires 250GB of memory and computation time is expected to be more than 6 days per timestep. For $|\vec{X}^t| = 6$ SIR obtained an approximate solutions to the online filtering problem in 12 seconds using 5 000 000 samples, and in 2 minutes when using 50 000 000 samples, and only requires storage for all samples. In fact, Figure 8 shows that the SIR algorithms scales linearly with the number of random variables in an ADBN (evaluated on 10 different models over 50 timesteps, each) and shows that approximate inference in large ADBNs is even feasible when considering the absolute runtime of the SIR algorithm.

7. Bayesian Network Semantics Revisited

For a large set of probabilistic graphical models, if not all, it is common to represent exactly one JPD, and, so far, situations, i.e., contexts from observations, have been discussed for ADBNs where as much partial structural information (deactive activator observations in $\vec{\epsilon}$) is evident to ensure the same in an ADBN. This is an intended property, as it allows an ADBN to remain freely instantiable and does not enforce constraints of regularity. This property is highly beneficial, since Motzek and Möller (2015a) show that not only acyclicity is a regularity constraint and all previously derived algorithms and problem discussions remain applicable for any discovered regularity conditions. Nevertheless, in a particular situation where insufficient observations can be made, this freedom of ADBNs introduces one drawback: one is forced to observe at least a minimal set of activators. In this section we show that by revisiting the semantics of a cyclic ADBN, one is able to remove this constraint completely s.t. no specific observation sets are enforced, one is allowed to observe every subset, and one still obtains a well-defined DPGM.

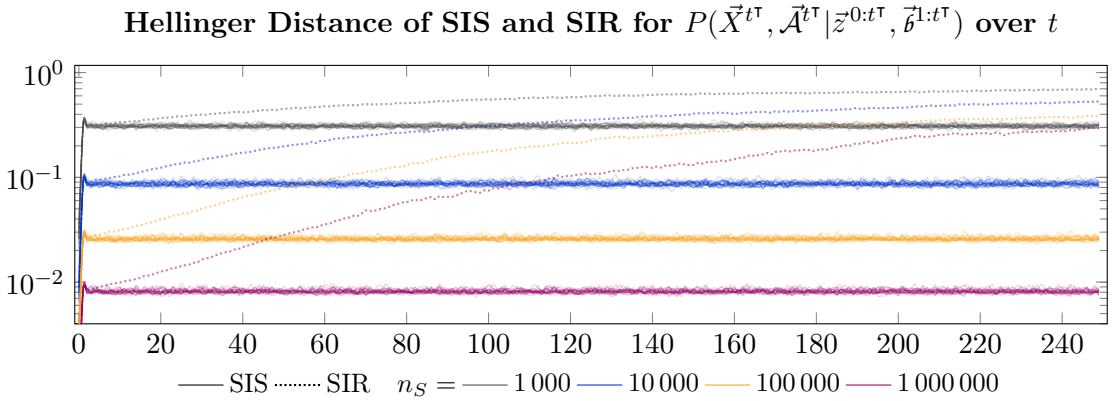

Figure 7: Stratified Sequential Importance Resampling (SIR) using n_S samples solves $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{i-1:i}, \vec{\delta}^{i-1:i}, i)$ at timestep i (abscissa) with constant and bounded error (Hellinger distance compared with exact inference, ordinate), whereas Sequential Importance Sampling (SIS) has exponential growth of error over time (*semilogarithmic plot*). SIR plotted overlain, mean plot for SIS.

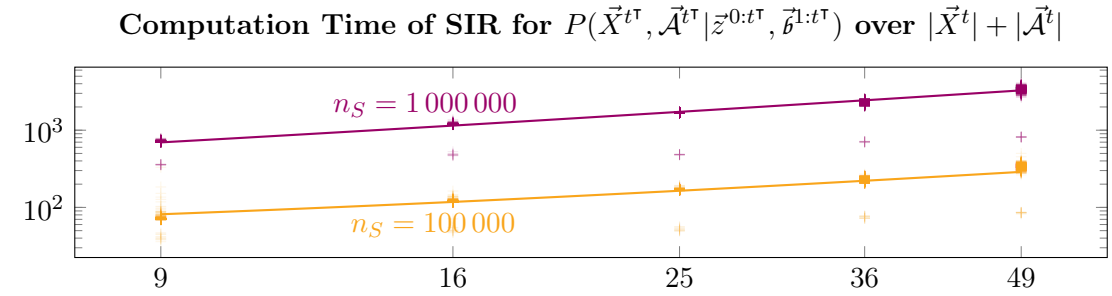

Figure 8: Stratified Sequential Importance Resampling (SIR) using n_S samples solves $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{t-1:t}, \vec{\delta}^{t-1:t}, t)$ (computation time in ms, ordinate) linearly in the number of random variables (abscissa) $N = |\vec{X}^t| + |\vec{\mathcal{A}}|$ (*double logarithmic plot*).

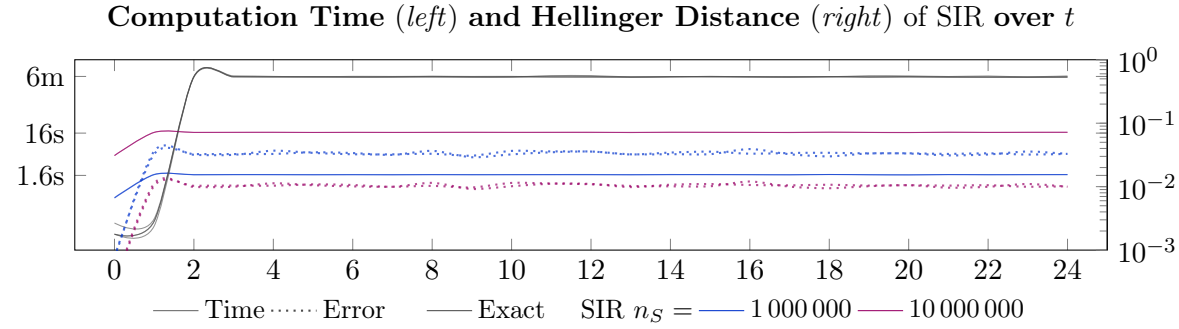

Figure 9: SIR solves $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{i-1:i}, \vec{\delta}^{i-1:i}, i)$ at timestep i (abscissa) magnitudes (*semilogarithmic plot*) faster (solid, left ordinate, computation time in ms) than an exact algorithm (black) and still achieves satisfying accuracy (dotted, right ordinate, Hellinger distance).

Under every regular observation $\vec{\ell}$, exactly one topological ordering (under a common lexicographical ordering) of an equivalent (dynamic) Bayesian network exists, and this topological ordering holds for all instantiations conforming with evidence, as shown in Proof A. In such a situation, inference is based on a single well-defined (dynamic) Bayesian network defining one joint probability distribution. If sufficient structural information are not evident from observations, multiple structures are represented by an ADBN, each of which defines a single joint probability distribution. In this section we move away from a dogma that a (D)BN represents solely one joint probability distribution, and, indeed, an ADBN is seen as an evolution of (D)BNs towards multi full joint probability distribution representations. Consequently, we show that *cyclic ADBNs are well-defined even if no observations are provided*.

If, in a specific context, Theorem 1 (and extensions) cannot be fulfilled, then an effective single structure for an equivalent DBN is not known, and an ADBN represents multiple joint probability distributions for every remaining, well-defined constellation of activator instantiations in the joint probability distribution. Note that no external constraints or conditions are given on the JPD by Proposition 4 and a single closed-form formula represents all possible joint probability distributions of DBNs without a need of a case-by-case consideration. In particular, we show that a cyclic ADBN is well-defined without introducing any external framework nor a novel calculus, such as those introduced, e.g., by Milch et al. (2005) with the $\lambda(\cdot)$ and comp$(\cdot)$ operators, or those introduced by Bilmes (2000) with the $z^t(q^t)$ operators. We revisit (A)DBN semantics such that not only a JPD is represented, but *multiple* JPDs. This is significantly different from considering every case separately, namely every single structure separately, as we consider a distribution over multiple possible structures, i.e., a distribution over multiple probabilistically overlapping distributions. We show that Proposition 4 remains universally applicable, i.e., one formula encapsulates all full joint distributions and, consecutively, every well-defined sub-JPD is given simply by an adequate subset.

7.1 Bayesian Networks of Bayesian Networks

In a situation in which sufficient structural information is not available, multiple regular and non-regular instantiations conform with evidence. Under the assumption that every timestep is regular, one knows that one of all remaining possible regular instantiations must be effective, over which one is able to specify a prior belief. This means that there exists a prior random distribution over all possible regular instantiations. Therefore, ADBNs are generalized towards extended ADBNs (eADBNs) in which no structural information need to be evident from data or observations. In eADBNs, multiple topological orderings (even under a common lexicographic ordering) exist, each of them belonging to one well-defined Bayesian network. Then, an eADBN represents a Bayesian network of Bayesian networks as depicted in Figure 10, or rather, represents a distribution over multiple well-defined joint probability distributions. An eADBN shares familiar syntax and semantics with classic Bayesian networks, without case-by-case analyses of possible structures or introduction of external constraints. An eADBN is formally defined as follows.

Definition 13 (Extended ADBN, eADBN)**.** An *eADBN* is defined as a tuple $(B_0, B_{\rightarrow})$ just like an ADBN (Definition 3). Additionally, random variables of a timestep t, i.e., $\vec{X}^t, A^t$, are seen as influenced by one (meta) random variable Ord^t. Let each value of Ord^t represent a possible obtainable topological ordering $ord_i^t \in \text{dom}(Ord^t)$ of random variables at time t under a common lexicographical ordering. Every *topological ordering* of random variables is obtainable by one minimal set of deactive activator instantiations (Proposition 3). Let $\vec{a}^t_{\blacklozenge i}$ be the minimal set of deactive activator instantiations to obtain a topological ordering at time t represented by ord_i^t. Then, let every CPD of activators $a^t \in \vec{a}^t_{\blacklozenge i}$ allocate the full probability mass at its deactive value, i.e., $\forall a^t \in \vec{a}^t_{\blacklozenge i} : P(\neg a^t | ord_i^t) = 1$. ▲

In an eADBN, additional random variables are identified (or introduced) to ADBNs and certain CPDs of activator follow special parameter settings. These special parameter settings enable one to perform inference in ADBNs without the need of minimal observation sets, as an eADBN is well-defined without constraints, as stated by the following theorem.

Theorem 7 (eADBN well-definedness)**.** An eADBN is well-defined for every instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}, \vec{ord}^{1:t}$ of all random variables $\vec{X}^{1:t}, A^{1:t}, Ord^{1:t}$. Semantics as $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}, Ord^{1:t\intercal})$ is well-defined and equivalent to DBN semantics given as the product of all locally defined CPDs. ▲

Under Theorem 7 an eADBN is well-defined for all observation and instantiations, i.e., no minimal observation sets need to be enforced. Therefore, one is able to reason about the structural identifying context itself, using classic Bayesian network inference as, e.g., shown in Section 5. Note that although cyclic dependencies are modeled and are not dissolved by activator contexts, neither external constraints nor global normalization factors need to be introduced, which still preserves the desired causal and local specifications, and the desired local interpretation of conditional probability distributions. Note further that not all possible structures need to be considered separately (e.g., by an external framework), but the consideration of all possible structure constellations is intrinsically handled by the semantics of eADBNs, i.e., one joint probability distribution handles all structural "cases."

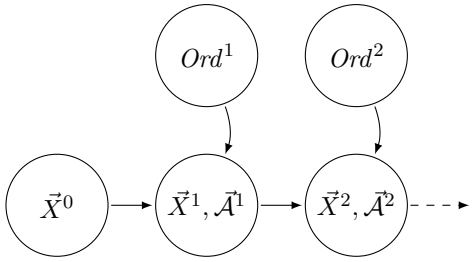

Figure 10: eADBNs are able to represent Bayesian network of Bayesian networks, i.e., represent a distribution of multiple well-defined probability distributions. Unlike ADBNs, eADBNs do not require any contextual, structural information. Specific parameter settings of $P(A^t_{XY} | Ord^t)$ allow for familiar Bayesian network syntax and semantics without requiring case-by-case analyses nor externally invoked frameworks.

Proof of Theorem 7 (eADBN well-definedness). For *every* instantiation $ord_i^{1:t}$ of $Ord^{1:t}$ there exists an equivalent instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ of random variables $\vec{X}^{0:t}, A^{1:t}$ that is enforced by extreme allocations of probability masses in all local conditional probability distributions. For every equivalent instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ there exists a topological ordering according to Theorem 1 and Proof A, and there exists a well-defined joint probability distribution $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal})$. Every well-defined joint probability distribution $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal})$ is seen as an entry $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal} | ord_i^{1:t})$ of a conditional probability distribution $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal} | Ord^{1:t})$. With a well-defined distribution over all possible orderings, i.e., $P(Ord^{1:t})$, a well-defined Bayesian network $\hat{B}$ is formed, defined by *two* random variables "$Ord^{1:t}$" and "$\vec{X}^{0:t}, A^{1:t}$," *one* prior probability distribution $P(Ord^{1:t})$ and *one* conditional probability distribution $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal} | Ord^{1:t})$ representing the joint probability distribution $P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}, \vec{Ord}^{1:t\intercal})$ as the product of all locally defined (C)PDs, i.e.,

$$P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}, Ord^{1:t\intercal}) = P(Ord^{1:t}) \cdot P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal} | Ord^{1:t}) \ .$$

Further, for an intra-timeslice dependency structure, $\hat{B}$ is seen as a dynamic Bayesian network $(\hat{B}_0, \hat{B}_\rightarrow)$ with two random variables per timeslice, as shown in Figure 10, where the global semantics of $(\hat{B}_0, \hat{B}_\rightarrow)$ is then given as

$$\begin{aligned} P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}, Ord^{1:t\intercal}) = P(\vec{X}^{0:t-1\intercal}, \vec{\mathcal{A}}^{1:t-1\intercal}, Ord^{1:t-1\intercal}) \\ \cdot P(Ord^t) \cdot P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal}, Ord^t) \ , \end{aligned}$$

and $(\hat{B}_0, \hat{B}_\rightarrow)$ is, for *all* instantiations, a well-defined DBN according to Proposition 1. □

Note that all derived approaches for exact inference are not based on certain well-definedness constraints but are solely based on the full joint probability distribution, which consists of a classical product of locally defined CPDs. This means that for inference, a well-definedness theorem is irrelevant and solely serves as a post-condition in order to ensure that results are correct. This circumstance is highly beneficial, as the presented well-definedness Theorem 1 only considers acyclicity, whereas Motzek and Möller (2015a) show that other regularity constraints exist beside acyclicity. As inference does not enforce any well-definedness, all derived approaches remain applicable. The same holds for eADBNs, even if cyclic dependencies are not resolved by observations. From an inference perspective, the only addition is a new random variable Ord^t, affecting some, or all, random variables of a timeslice without causing any cycles. Further, one could define activator random variables for every dependence of Ord^t to obtain a classical intra-timeslice ADBN, for which all derived approaches remain applicable.

The following example shows how the running example is extended by a meta random variable, and then supports even an unobserved context of message transfer variables.

Example 9 (Regulatory compliance in eADBNs)**.** *So far, at every timestep at least as many message transfers must be observed to ensure regularity. Namely, at least $n^2/2$ message transfer variables M_{XY}^t have to be observed to be deactive for obtaining a well-defined ADBN. With Definition 13 all queries to an eADBN are answerable without any contextual information about message transfers. Therefore, an eADBN also fully supports predictive queries. We*

demonstrate the parametrization of $\mathrm{dom}(Ord^t)$ *and* $P(Ord^t)$ *and a respective parametrization of activator random variables as an example for message transfer random variable* M^t_{CD}.

In the running example with employees Claire, Don and Earl, one obtains six topological orderings under a common lexicographical ordering. Namely, one obtains $C \succ D \succ E, C \succ E \succ D, D \succ C \succ E, D \succ E \succ C, E \succ D \succ C, E \succ C \succ D$*, which resemble the domain of* Ord^t*, abbreviated as* $\mathrm{dom}(Ord^t) = \langle cde, ced, dce, dec, edc, ecd\rangle$*. For every value of* $\mathrm{dom}(Ord^t)$ *one is now able to specify a prior belief over every topological ordering as shown in Table 2. For example, if Claire is a supervisor of Don and Earl, topological orderings starting with* C *might be more likely, representing that it is more likely that the supervisor influences her employees and sends tainted messages to them. As stated by Proposition 3 every topological ordering can be induced by an instantiation of activator random variables, which are influenced by* Ord^t *in an eADBN. Considering* M^t_{CD}*, an adequate parameter setting of* $P(M^t_{CD}|Ord^t)$ *is given in Table 3, where the topological orderings* $dce \in \mathrm{dom}(Ord^t)$ *(i.e.,* $D \succ C \succ E$*) and* $edc \in \mathrm{dom}(Ord^t)$ *(i.e.,* $E \succ D \succ C$*) must enforce an allocation of full probability mass on* $\neg m^t_{CD}$.

Table 2: Example for a distribution of $P(Ord^t)$ for the running example with arbitrary numbers α—γ with $\alpha + \beta + \chi + \delta + \varepsilon + \gamma = 1$. A numerical example instantiation is given, resembling that a topological ordering starting with C is more likely, e.g., that Claire is a supervisor of Don and Earl. Generation of $\mathrm{dom}(Ord^t)$ and $P(Ord^t)$ is straightforward and minimally invasive to the running example.

Ord^t	$\mathbf{P}(Ord^t)$	Ord^t	$\mathbf{P}(Ord^t)$
cde	$\alpha = 0.3$	dec	$\delta = 0.1$
ced	$\beta = 0.3$	edc	$\varepsilon = 0.1$
dce	$\chi = 0.1$	ecd	$\gamma = 0.1$

Table 3: Example for a CPD of $P(M^t_{CD}|Ord^t)$ for the running example with arbitrary numbers ι—μ. A numerical example instantiation is given based on an extension of the previous parameter $P(+m^t_{CD}) = \eta = 0.5$ and shows that a generation of $P(+m^t_{CD}|Ord^t)$ is straightforward and minimally invasive to the running example.

Ord^t	$\mathbf{P}(+m^t_{CD}\|Ord^t)$	Ord^t	$\mathbf{P}(+m^t_{CD}\|Ord^t)$
cde	$\iota = \eta$	dec	$\lambda = \eta$
ced	$\kappa = \eta$	edc	0
dce	0	ecd	$\mu = \eta$

As we have demonstrated in this article, (cyclic) ADBNs naturally arise in domains by considering direct dependencies from a local point of view. Not always a meta variable Ord^t is immediately part of a domain and Ord^t must be introduced to ensure regularity in case sufficient structural information is not evident from observations. Still, the example shows

that an introduction is straightforward, and is minimally invasive, as only CPDs of activator random variables must be modified in a simple schema. If intended, an extension of an ADBN towards an eADBN can be implemented automatically and an Ord^t variable need not be made explicit if an inference engine is able to distinguish cases of insufficient structural information. Nevertheless, making Ord^t explicit has the benefit that a sufficiency check of regularity is not required and any Bayesian inference engine is intrinsically able to handle even a cyclic ADBN.

In an eADBN, activator random variables need not necessarily exist as independent random variables, as a topological ordering identifying random variable is sufficient. Still, the following remark shows that it is desirable to maintain individual activator random variables.

Remark 4 (Collapsed activator random variables)**.** *If a topological-ordering-identifying random variable (Ord) is present in a domain, activator random variables A_{ij}^t can collapse with Ord^t, i.e., Ord^t represents each and every A_{ij}^{st}. While this leads to a well-defined A(D)BN, it introduces a significant modeling overhead, as not all activator random variables are relevant for one X_i^t, i.e., large subsets of* $\mathrm{dom}(Ord^t)$ *identify the same parameter in a local CPD of X_i^t. Only random variables included in the vectors $\vec{A}_i^{st}, s \leq t$ are relevant for X_i^t. For example, in a dense intra-timeslice ADBN with n state variables, one obtains $n!$ topological orderings of state variables. If activator random variables are collapsed with Ord^t, then every CPD specification of a state variable X_i^t must consider $n!$ cases of* $\mathrm{dom}(Ord^t)$*. With explicit activator random variables, every CPD specification of a state variable X_i^t must solely consider 2^{n-1} cases of* $\mathrm{dom}(\vec{A}_i^t)$*.*

Still, for one cyclic dependency, i.e., two potential topological orderings and two activator random variables, no overhead is introduced as shown in the following section as a continuation of Example 1 regarding the *Joke-Cry*-domain. As mentioned in the beginning, the *Joke-Cry* domain, in fact, does represent an eA(D)BN.

7.2 Extended Activator Bayesian Networks

We motivate the identification and exploitation of activator random variables in *dynamic* BNs, as they are often associated with cyclic dependencies and—from another perspective—for reasoning over time. Of course, also static "time-less" Bayesian networks benefit from activator conditions of random variables in order to model processes where certain dependencies change depending on other contexts. If roles of causes and effects are context-specific then cyclic dependencies are needed, which are permitted in activator Bayesian networks, which are defined as follows.

Definition 14 (Activator Bayesian networks, ABN)**.** Unrolling the first two timeslices of a cyclic ADBN yields in a classic BN with cyclic dependencies and activators, which is defined as an *ABN*. A cyclic ABN represents a static Bayesian network in which local structures are not known in advance and dependent on specific contexts. This is beneficial for Bayesian networks in which roles of cause and effects are not uniquely identifiable, and in which dependencies are sensitive to a context. An *ABN's semantics* is well-defined by the unrolled ADBN's semantics as defined in Theorem 1 or in Theorem 7. For the case that a topological-ordering-identifying random variable is present in an ABN, i.e., an ABN's semantics is well-defined under Theorem 7, we speak of an *extended ABN* (eABN). ▲

Context-specific influence directions are often found in human emotions caused by mutual interaction and are required to model a causally correct knowledge base for an artificial intelligence as the introductory Example 1 shows. In fact, the *Joke-Cry* domain from Example 1 represents an example for an extended ADBN (Definition 13) as well as an eABN (Definition 14). The following example shortly repeats the domain and demonstrates the modeling approach as an eABN.

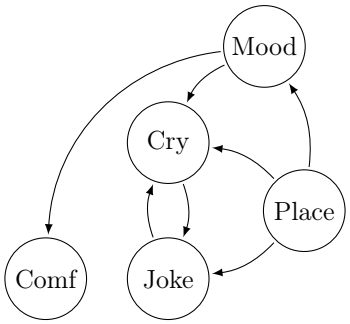

Figure 11: Modeling causal dependencies often requires cyclic dependencies. If one is supposed to reason over *Cry*ing, potential influences *of Joke* or *on Joke* must be considered. Without a link between *Joke* and *Cry*, the only possible explanation for crying at a party is that the person is *Mood* = *sad* or a person always cries at parties—usually a wrong assumption. And without a link from Cry to Joke, the only explanation for a Joke being told at a funeral is the funeral itself—usually a quite macabre assumption. Repeated Figure 1 from Page 4.

Example 10 (Extended activator Bayesian network)**.** *To repeat, people might start crying from laughter because of hearing a (very) good joke. However, assuming only this causal direction at, e.g., a funeral, is a macabre assumption—the reasons why jokes are possibly told at a funeral are to possibly cheer up sad and crying people. As discussed in the caption of Figure 11 only modeling one causal direction is not an option.*

A cyclic dependency between both random variables Cry and Joke in Figure 11 is causally required in order to capture a true causality between both random variables, whose direction is only identifiable in a context of another random variable Place. Considering Figure 11 as an extended eABN, Place represents meta-random variable Ord. In fact, in this example Ord is not a meta-variable, but is part of the domain: Seeing the domain of Place as cry − worthy places like a funeral or wedding, and usually happy places like a festival party or comedyclub, directly represents all possible topological orderings of the graph. Note that activators $A_{Cry\,Joke}$ *and* $A_{Joke\,Cry}$ *are collapsed with Place in this example.*

By modeling the domain as an eABN, one includes all desired influences and dependencies as seen from a world-representation point of view: One can cry from happiness and from hearing very good jokes in the surroundings of happy, comfortable places, but people tell jokes to warm-up a chilling atmosphere among crying, sad people at darker places. Naturally, the cyclic eABN, as shown in Figure 11, also covers that at a happy place, a mood is usually happy, from which it is less likely to cry and it is more likely that a good joke was told. With an eABN all possible relationships are included in one general model, and it is well-defined

even if the context of Place is unobserved, i.e., no single causality is identifiable. As all potential models are intrinsically represented by one eABN, one is able to reason about Place—the structural identifying context—as well, and, effectively, inference is based on all possible models at the same time.

In this example, two full joint probability distributions are modeled and embedded over which a distribution ($P(Place)$) is modeled. The semantics of this eADBN is a distribution based on the prior random distribution of Place over a full joint probability distribution over all random variables Place, Cry, Joke, Mood.

This example not only shows that activators exist naturally in domains, but also demonstrates that topological ordering identifying random variables occur from local, causal modeling approaches, as well. As shown in previous sections, Example 10 represents a well-defined probabilistic graphical model, neither requiring external reasoning frameworks to handle every specific context of *Place* separately, nor requiring global normalization factors due to cyclic graph structures. We have shown that (i) various algorithms for exact and approximate inference and learning of A(D)BNs follow familiar schemes and (ii) the identification of activator random variables does not introduce any significant overhead. Further, this section clarifies that not even restrictions on instantiations or observations are enforced on A(D)BNs by adequate modeling techniques. Considering all of above, we feel confident to say that (dynamic) Bayesian networks can be based on cyclic graphs.

8. Discussion and Related Work

Using an ADBN has the benefit of anticipating indirect causes per timestep in an over-the-time evolving process and allows for modeling cyclic dependencies from a local point of view. Still, some random variables need to be identified as activator random variables, on which minimal observation sets are enforced (Theorem 1: any acyclic constellation of probably active activators is allowed) or require extended modeling approaches (see Definition 13). However, we come from a point of view where activators exist naturally in the modeled domain, and, in fact, Proposition 2 shows that classic (diagonal) DBNs are significantly restricted as well. We conclude that cyclic, as well as, diagonal ADBNs are only usable for certain instantiations of all random variables. To emphasize that the restrictions on cyclic ADBNs are far smaller than the identified restrictions on diagonal ADBNs, we explicitly compare the numbers of allowed instantiations in cyclic (Theorem 1) and diagonal ADBNs (Proposition 2) per timestep. Note that any cyclic instantiation prohibited in a cyclic ADBN includes the necessity to anticipate indirect influences, i.e., is prohibited in diagonal DBNs as well.

Proposition 8 (Number of indirection-free instantiations in diagonal ADBNs)**.** Proposition 2 enforces that instantiations $(\vec{x}^{0:t}, \vec{a}^{1:t})$ of $(\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t})$ of a dense diagonal (A)DBN cannot contain two possibly active activator instantiations a^t_{*i}, a^t_{i*} to obtain an intended joint probability $P(\vec{x}^{0:t\intercal}, \vec{a}^{1:t\intercal})$ result. Thus, the set of active activators in an instantiation $\vec{a}^t$ must form a uniformly directed bipartite graph, where isolated nodes belong to a fixed group. For $n = |\vec{X}^t|$ state variables, the number of these bipartite digraphs is given in Sequence $A001831$ by Sloane (2015). For every allowed activator constellation $\vec{a}^t$, 2^n $\vec{X}^t$-instantiations are possible. The total number $\mathcal{N}'_n$ of regular instantiations of a joint probability in a dense

inter-timeslice (A)DBN with n state variables is therefore

$$\mathcal{N}'_n = 2^n \cdot \sum_{k=0}^{n} \binom{n}{k} \cdot \left(2^k - 1\right)^{n-k} \qquad \blacktriangle$$

Proposition 9 (Number of regular instantiations in cyclic ADBNs)**.** Theorem 1 enforces that instantiations $(\vec{x}^{0:t}, \vec{a}^{1:t})$ of $(\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t})$ do not contain a cycle w.r.t. active activators to obtain a well-defined joint probability $P(\vec{x}^{0:t\intercal}, \vec{a}^{1:t\intercal})$ result. Thus, the set of active activators in an instantiation $\vec{a}^t$ must form a directed acyclic graph (DAG), where every active activator $a^t_{ij} \in \vec{a}^t$ represents an edge from a node i to j in the DAG. The number of DAGs for $n = |\vec{X}^t|$ nodes is given in Sequence $A003024$ by Sloane (2015). For every allowed activator constellation $\vec{a}^t$, 2^n $\vec{X}^t$-instantiations are possible. The total number $\mathcal{N}^{\mathcal{O}}{}_n$ of regular instantiations of a joint probability in a dense intra-timeslice DBN with n state variables is therefore

$$\mathcal{N}^{\mathcal{O}}{}_n = 2^n \cdot \mathrm{A003024}_n \qquad \blacktriangle$$

Figure 12 shows both curves of Proposition 8 and Proposition 9. Even in a logarithmic plot, a cyclic ADBN has an exponential advantage in favor of a classic diagonal (A)DBN when considering the number of allowed instantiations per timestep.

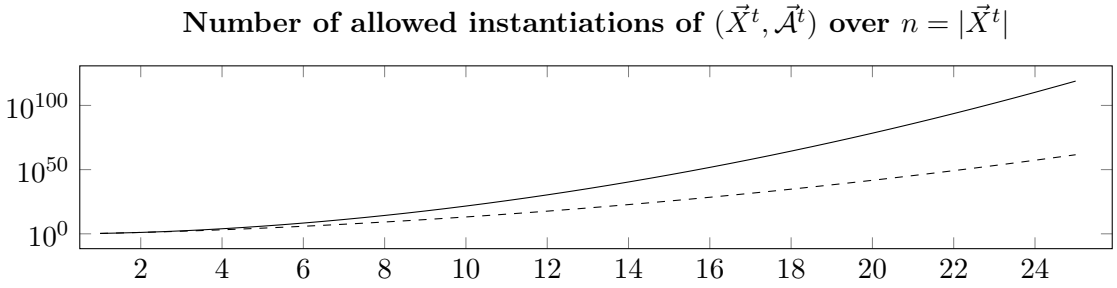

Figure 12: Cyclic ADBNs ($\mathcal{N}^{\mathcal{O}}{}_n$, solid, Proposition 9) clearly outperform classic diagonal DBNs ($\mathcal{N}'_n$, dashed, Proposition 8) in the number of allowed instantiations of $(\vec{X}^t, \vec{\mathcal{A}}^t)$. (*semi-logarithmic plot*) per timestep.

The need to anticipate indirect influences originates from coarse observation timesteps, where indirect influences must be anticipated to explain observations. While we motivate coarse timesteps from an unavailability of observations at a finer time-scale, the choice of coarser timesteps is also motivated by computational feasibility. Not being bound to the finest available observation granularity relaxes the rate of needed time-updates and motivates Asynchronous DBNs by Pfeffer and Tai (2005) using Nodelman, Shelton, and Koller's (2002) Continuous-Time-BNs (CTBNs). Asynchronous DBNs allow for a distributed, decentralized update of nodes in a DBN, instead of enforcing a synchronized update of all nodes in a DBN at the smallest given update frequency. Still, Asynchronous DBNs and CTBNs run into the same problem given in Proposition 2 of anticipating indirect influences during one timestep, if observations are not known at a near continuous time granularity. Our work considers dynamic Bayesian networks as a stationary process over time, where random variables and dependencies between them represent a causal influence, and a time represents an actual

flow of time. We consider random variables with discrete domains and consider all random variables to be potentially observable. We pointed out causal discrepancies in DBNs, if timeslices are not infinitesimal small. The same view on dynamic Bayesian networks is taken by Sanghai et al. (2005) in their work on relational DBNs and by Jaeger (2001) in his work on recursive relation probabilistic models. Both works follow the intention of Pearl and Russell (2003) to model causal relationships instead of reasoning processes, and even extend DBNs to relational domains. Both, Jaeger and Sanghai et al., consider timesteps at an infinitesimal small time granularity, under which only single events occur per timestep, by which both circumnavigate the problem of anticipating indirect influences under an acyclicity constraint of relations. Nevertheless, it remains an interesting future work to integrate an ADBN's adapting structure per timestep into relational domains.

Throughout this article, cyclic ADBNs are compared with diagonal ADBN and it is shown that transforming cyclic ADBNs to diagonal ADBNs by following a naive "resolving cycles over time"-approach delivers highly spurious results. As discussed in Section 4 another form of diagonal inter-timeslice ADBNs exists: a *hybrid* model consisting of inter- and intra-timeslice dependencies, where as much intra-timeslice dependencies are modeled as possible without creating cyclces, and remaining dependencies are bent to a consecutive timestep. We refrained from discussing this form of a model throughout this article, as it is a completely arbitrary decision why some dependencies are modeled as inter- and some are modeled as intra-timeslice dependencies. Nevertheless, from a reasoning perspective, a desired influence between every random variable is present in each timeslice. However, these influences are then modeled from a completely acausal perspective as dependencies then represent cause→effect as well as effect→cause relationships at the same time. Due to this acausality and inconsequence, the parametrization of hybrid models becomes practically impossible for a human, and parameters, i.e., CPDs, do not provide any local or intuitive interpretation. Such hybrid acausal models then require a spurious inverse-reasoning mechanism over the complete model for specification and interpretation. Hence, we believe that such hybrid models may only be of use in black-box approaches, where an underlying model is learned by a framework but not intended to be interpreted by a human. Motzek (2016) introduces a learning approach for ADBNs while considering activator constraints and their semantic implications on learning of ADBNs and DBNs. Noteworthy, Motzek (2016) shows that a modification of an EM approach remains applicable towards learning cyclic ADBNs without introducing an external structure-analysis nor overhead, and remains applicable even if activator instantiations are missing from data, i.e., even if activators are uninstantiated, which affects the underlying structure, which, in turn, is required for restoring these missing instantiations. Further, they compare the inference accuracy of learned models for various forms of cyclic, diagonal and hybrid ADBNs. In fact, they show that learned cyclic ADBNs deliver highly accurate results, but that neither fully diagonal (i.e., dense inter-timeslice) nor hybrid (i.e., mixture of intra- and inter-timeslice) models are learnable that provide any accurate inference results.

In essence, ADBNs change their structure at every timestep depending on instantiations of activators. DBNs with changing structure over time are also the main focus of non-stationary DBNs by Robinson and Hartemink (2008) and of time-varying DBNs by Song, Kolar, and Xing (2009). However, Robinson and Hartemink and Song et al. take a fundamentally different perspective on changing structures, where changing structures represent an evolutionary change of a structure over time during a long evolutionary process and structures are slowly

modified by a set of possible actions. ADBNs can rapidly change over time, namely at *every* timestep depending on a specific context.In fact, ADBNs succeed non-stationary DBNs in their expressiveness by capturing all possible structures in one ADBN where the context, i.e., $\mathcal{A}^t$ and Ord^t, specifies a needed substructure for each evolutionary step. By discretizing the evolution of a structure in a non-stationary DBN into discrete epochs with constant structure, i.e., one introduces a new epoch after every structure modification, one obtains a random variable $Epoch^t$ depending on $Epoch^{t-1}$ (resembling the evolutionary character), which immediately represents an Ord^t and an implied activator constellation in an eADBN (compare Definition 13). Therefore, every non-stationary DBN is representable as an eADBN in well-defined and classical DBN syntax and semantics.

Fundamental to our work is the introduction of activator random variables, which exploit context-specific independencies in Bayesian networks. The presence of context-specific independencies and their advantages for easier specifications and representations of local parameters have notably been described by Boutilier et al. (1996). Poole and Zhang (2003) extend work by Boutilier et al. in order to exploit context-specific independencies for more efficient exact reasoning in (dynamic) Bayesian networks by considering effective independencies in instantiated structures. Notwithstanding, their work was fundamental for our work, but the authors solely focused on performance optimization, whereas we focus on increasing expressiveness of DBNs for anticipating indirect influences correctly by rapid adaptations to (even unknown) contexts.

Dynamic Bayesian networks in which actual dependencies depend on a specific context of random variables are a main focus of Dynamic Bayesian Multinets (DBMs) by Bilmes (2000) and represent a form of dynamic Bayesian networks similar to our work. However, in a DBM a value of a single externally introduced variable (Q) steers a structure of a network by introducing a new syntax using so-called dependency functions. Encoding every possible structure in one variable introduces a significant overhead in specifications and veers away from a world-representing declaration. In ADBNs multiple activator variables are part of a domain and, thus, do not introduce a modeling overhead. Further, activator random variables adhere all properties of classic random variables neither introducing a new syntax nor a new semantics. On top of that, in an ADBN a higher degree of expressiveness is achieved, as activators can be left unobserved, representing that part of a specific structure is left unknown. Nevertheless, any DBM is transformable into an ADBN, but not vice versa, as DBMs explicitly do not support proactively designed cyclic dependencies and run into problems involving indirect influences as discussed in Proprosition 2.

A further consideration of roles and implications of context-specific independencies in probabilistic graphical models (PGMs) is done by Milch et al. (2005) focusing on an increased expressiveness of PGMs by the framework of (infinite) contingent Bayesian networks (CBNs). In CBNs edges are labeled with instantiations of some random variables, if the edge, i.e., dependency, is subject to a context-specific change. This edge-labeling follows a similar motivation as activator random variables do, and, similarly, an ADBN-activator-random-variable-instantiation $A_{XY} = {}^{+}a_{xy}$ can be seen as such an edge label in CBNs. Most notably, Milch et al. identify that certain domains cannot be modeled as one acyclic PGM and cyclic dependencies are required as we have motivated as well by the taintedness domain. However, Milch et al., similarly to Geiger and Heckerman (1996), introduce a novel calculus for their context-specific PGMs, which stands in a significant contrast to ADBNs. In ADBNs no novel

calculus is required, no external "outerloop" is required, and all random variables, CPDs, and the JPD as the product of all locally defined CPDs are business as usual. Moreover, CBNs make acyclicity constraints explicit by exclusive edge-labels, i.e., it is impossible to instantiate a (unrolled) cyclic CBN, whereas, in the ADBN formalism, a cyclic ADBN may be instantiated/unrolled, as regularity constraints are never enforced or required to become explicit. As Motzek and Möller (2015a) show, this circumstance is highly beneficial, as not only acyclicity is a regularity constraint. Furthermore, an explicit representation of all regularity constraints of an edge would introduce a significantly high modeling overhead of the taintedness domain in a CBN, where multiple context-specific cycles exist, as all potential instantiation-combinations for forming a directed-acyclic-graph would need to be made explicit along each edge, which cannot provide an intuitive and local design of such models. Moreover, we have shown that classical algorithms such as the forward-backward algorithm remain applicable for solving filtering or smoothing problems in ADBNs without any significant modification. While CBNs may be able to represent ADBNs to some extent, it remains unclear whether the novelly introduced CBN calculus still allows these algorithms to persist and remain applicable.

Our proof that an ADBN template $B_{\rightarrow}$ can be based on a cyclic graph under certain conditions is based on an equivalence of an unrolled dynamic Bayesian network whose instantiations are equivalent to instantiations of an acyclic dynamic Bayesian network, and allows that ADBNs are designed with cyclic structures proactively, as they are first resolvable in an instantiation and, in fact, represents more structures with which observations are explainable. Sets of equivalent classes of graphs are also considered by Acid and de Campos (2003) for more efficient structure learning in Bayesian networks. Learning ADBNs represents a novel research area of Bayesian network learning, as a structure is not known in advance and can only be designed, i.e., learned, proactively. Nevertheless, data as, e.g., created by process as in the running example, are inexplicable, i.e., unlearnable without an equivalent ADBN. As Proposition 3 shows, a dense inter-timeslice ADBN contains all possible inter-timeslice ADBNs and equivalent structures must be considered during ADBN learning. As an ADBN structure does not only change slowly over time, and, namely, requires a rapid context-specific structure change at every timestep, learning approaches as presented by Robinson and Hartemink (2010) for non-stationary DBNs are not immediately applicable.

A view on DBNs as, e.g., taken by us, is significantly more expressive than viewing DBNs as an unrolled hidden Markov model, where specific variables must be constantly observed and a view over "time" is used to resolve cycles, instead of representing a wall-clock time. Notwithstanding, if cyclic models are intended to be designed, one could switch to chain graphs as described by Drton (2009) or general Markov networks. However, by switching to (partially) undirected models one loses the direct and intuitive interpretation of locally specified conditional probability distributions by introducing a burden of computing non-local normalization quotients. In cyclic (e)ADBNs, all desired local semantics are preserved without introducing any computational or modeling overhead. On top of that, an undirected model inherently models a different domain as undirected edges imply symmetric influences, whereas in an ADBN we can specify asymmetric influence strengths. Further, a cyclic Markov network models a steady state influence domain, where influences let random variables converge to a stable state. The latter is an application of a diagonal DBN, where a cyclic directed graphical model is unfolded into a diagonal DBN and is simulated over a long timeperiod to obtain

aforesaid stable state. Note that, the "misuse" of time to simulate this cyclic behavior is exactly the problem that motivated this article and leads to the problem of being unable to anticipate indirect influences in one timestep of a DBN, if time is supposed to represent an actual wall-clock time, as envisioned in ADBNs. Nevertheless, as Proposition 3 shows, an ADBN can model any DBN and can therefore also be used to simulate feedback loops, i.e., cyclic dependencies that do not resolve to be acyclic.

Recent advances in probabilistic logic languages by Fierens et al. (2015) and de Raedt et al. (2007) allow for encoding and simulating (dynamic) Bayesian networks using probabilistic rules. Often probabilistic logic languages are not subject to acyclicity constraints as a reasoning process is not based on Bayesian inference, but are reduced to other commonly known computational problems. This means that similar answers to inference queries in ADBNs are obtainable in, e.g., works by de Raedt et al. by adequate specifications of probabilistic rules, but rules for reasoning are designed instead of representing a causal process as a DBN.

We have discussed the complexity of problems over time and have shown that in an ADBN one obtains the same, and even simpler (see Example 7) complexities as in classic DBNs. However, like in any other DBN, the complexity in terms of number of random variables of query problems demands approximate inference techniques. We have shown that approximate inference techniques renders finding answers to inference problems in ADBNs tractable, even in large-scale domains. The discussed sequential importance sampling approach requires that at every timestep an effective topological ordering is made explicitly, i.e., a topological ordering must be obtained from made observations. Performing a topological sorting does not represent a bottleneck and the requirement does not come as a surprise, as sequential importance sampling is an adaptation of likelihood weighting, which does require a topological ordering similar to prior and rejection sampling (cf., Russell & Norvig, 2010, pp. 530–535). Considering that Gibbs sampling is an approximate inference approach for Bayesian network that is not based on a topological ordering and is only based local conditional independencies, an adaption of Gibbs sampling towards approximate inference in dynamic (activator) Bayesian networks is a promising research field for future work. Moreover, the combination of exact and approximate inference in ADBNs represent a highly interesting future research area: If substructures are formed for which exact inference remains tractable, exact inference can be combined with approximate inference for the remainder of the network. As shown by Doucet et al. (2000) in the form of Rao-Blackwellised Particle Filtering (RBPF) such approaches deliver more accurate inference results than, e.g., a classical SIR procedure. While in classical DBNs a structure remains constant and a "tractable substructure" is known in advance, adapting RBPF to ADBNs motivates a new form of RBPF: substructural-adaptive RBPF.

Notwithstanding, it is possible that an observation is not regular in a particular situation, if, e.g., a time granularity is chosen too coarse or too few observations are acquired. Still, Theorem 1 now represents a direct indicator for potentially spurious results, and Section 7 has introduced a solution to insufficient structural context information. In a situation where a context is actually observed to be cyclic, i.e., too many active activators are observed, an action to ensure well-definedness is needed, but is beyond the scope of this article. Such an action would be, for example, to move small subsets of observations to a neighboring timestep or to find a most likely, but minimally invasive alternative observation that conforms to Theorem 1.

9. Conclusion

We have shown that indirect causes in dynamic Bayesian networks cause conflicts in representing causality by which highly spurious results are returned. These conflicts arise from using a modeled dimension, i.e., time as a wall-clock time, for assuring syntactic requirements. By identifying activator criteria of random variables in (dynamic) Bayesian networks, we introduce ADBNs and are able to move acyclicity constraints from a design phase to a later operation phase. Without the need of external reasoning frameworks, degrading a Bayesian network to a reasoning process, we obtained a solid mathematical basis sound to Bayesian networks with a causally correct anticipation of indirect causes in dynamic Bayesian networks. ADBNs provide the ability to intrinsically let a DBN adapt to observed contexts, where a DBN's structure is not known in advance and changes over time while maintaining a Bayesian network as a world-representing first-class declaration. Most importantly, we show that ADBNs are able to be based on cyclic graphs while remaining in a classical calculus with classical random variables and classical CPDs preserving desired local semantics, without introducing neither new procedural inference approaches, nor any new calculus operators.

Future work is dedicated to integrate new and varying acyclicity constraints when considering properties of local CPDs as done by Motzek and Möller (2015a) into new approximate inference techniques and extensions of ADBNs towards intrinsic representations of relational domains in a classical (A)DBN formalism.

Appendix A. Proof of Well-Definedness

We prove Theorem 1 of the well-definedness of a (cyclic) ADBN by showing that Proposition 4, i.e., the joint probability over all random variables of a dense intra-timeslice ADBN, corresponds to a joint probability of a well-defined DBN (according to Proposition 1) under Theorem 1. This is a proof based on one special ADBN, but Proposition 3 states that a dense intra-timeslice ADBN includes all possible intra-timeslice dependencies and thus the following proof is a proof for Theorem 1. An ADBN can include inter-timeslice dependencies, but these are subject to the well-definedness Proposition 1 (see Theorem 1) and the following proof is equivalent.

Notation 7 (Vector probability operands)**.** *For brevity, we define a probability of a vector containing random variables as a shorthand notation for a product of probabilities. Let* $\vec{X}$*,* $\vec{Z}$*,* $|\vec{X}| = |\vec{Z}|$ *be column vectors of random variables. Let* $\mathbf{P}_\Gamma(\vec{X}|Y,\vec{Z})$ *denote the product of probabilities* $P(X_i|Y,Z_i)$ *where* X_i *and* Z_i *are taken row-wise from* $\vec{X}$ *and* $\vec{Z}$*, except rows identified in the exclusion-set* Γ*. Scalars* Y *are repeated in every row. Formally,*

$$\mathbf{P}_\Gamma(\vec{X}|Y,\vec{Z}) = \prod_i P(X_i|Y,Z_i) \qquad i \in \{1 \leq i \leq |\vec{X}|\} \backslash \Gamma$$

Respectively, we apply this notation to (conditional) probabilities with n-ary dependencies and without dependencies, i.e., prior random variables.

Notation 8 (Lexicographic order)**.** *Let* $\prec$ *be a lexicographic term order, such that* $X_*^{t-1} \prec X_*^t$*,* $X_i^t \prec X_{i+1}^t$*, and* $A_{**}^{t-1} \prec A_{**}^t$*,* $A_{i*}^t \prec A_{(i+1)*}^t$*,* $A_{ij}^t \prec A_{i(j+1)}^t$*, and* $A_{**}^t \prec X_*^t$*,* $X_*^{t-1} \prec A_{**}^t$*.*

We rewrite Proposition 4 using Notation 7 as:

Proposition 10 (Shorthand joint probability distribution notation)**.** From Proposition 4, a dense intra-timeslice ADBN's semantics is

$$P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) = \prod_{X_k^0 \in \vec{X}^0} P(X_k^0) \cdot \prod_{i=1}^{t} \prod_{X_k^i \in \vec{X}^i} P(X_k^i | \vec{X}^{i\intercal} \backslash X_k^i, \vec{A}_k^{i\intercal}, X_k^{i-1}) \cdot \prod_{A_{cv}^i \in \vec{\mathcal{A}}^i} P(A_{cv}^i) \ ,$$

written for brevity using Notation 7 as

$$P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) = \mathbf{P}(\vec{X}^0) \cdot \prod_{i=1}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \qquad \blacktriangle$$

The shorthand allows for shorter notation of products where some indexes of a product need to be excluded.

Proof of Theorem 1 (ADBN well-definedness). We show that the joint probability distribution stated in Proposition 4 is indeed well-defined for every instantiation of $(\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t})$, if for all t the instantiation $\vec{a}^t$ of $\vec{\mathcal{A}}^t$ follows Eq. 3 by Theorem 1. We show this by reversing conditional independency assumptions in the semantic joint probability and that one obtains a topological ordering of a syntactical graph structure, i.e., a syntactically cyclic graph structure $B_{\rightarrow}$ is semantically a DAG for which Bayesian network semantics defines the same joint probability distribution as defined under restrictions in Theorem 1. B_0 can be written as

$$P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) = P(X_1^0) \cdot \ldots \cdot P(X_n^0) \cdot \gamma = P(X_1^0, \ldots, X_n^0) \cdot \gamma = P(\vec{X}^{0\intercal}) \cdot \gamma \ ,$$

with

$$\gamma = \prod_{i=1}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ .$$

Consecutively, one is able to roll up the joint distribution according to Bayes' chain rule. Considering an extreme case of a set of activators corresponding to Eq. 3, it is straightforward that in any instantiation following Eq. 3 it must always hold that $\exists X_{E1}^1 : \forall i A_{i(E1)}^1 = \mathit{false}$, such that due to Eq. 1, the set of activators and previous states uniquely identify the CPD entry of $P(X_{E1}^1 | \ldots)$ and X_{E1}^1 becomes independent of all other $\vec{X}^1$, such that the joint probability can be written as

$$P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) = P(\vec{X}^{0\intercal}) \cdot P(X_{E1}^1 | *, \vec{A}_{E1}^{1\intercal}, X_{E1}^0) \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{1\intercal}, \vec{X}^0) \cdot \mathbf{P}(\vec{\mathcal{A}}^1) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \quad (10)$$

By reversing X_{E1}^1's conditional independence assumption, i.e., one reverses "if X is independent of Z then $P(X|Y,Z) = P(X|Y)$" and enriches $P(X|Y)$ with conditionally independent random variable(s) Z to $P(X|Y,Z)$, one obtains

$$P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) = P(\vec{X}^{0\intercal}) \cdot P(X_{E1}^1 | *, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot \mathbf{P}(\vec{\mathcal{A}}^1) \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1 | \vec{X}^{1\intercal} \backslash \vec{X}^1, A^{1\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i\intercal} \backslash \vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ .$$

Hence, with

$$\begin{aligned}\mathbf{P}(\vec{\mathcal{A}}^t) &= P(A^t_{12}) \cdot \ldots \cdot P(A^t_{1n}) \cdot \ldots \cdot P(A^t_{n1}) \cdot \ldots \cdot P(A^t_{n(n-1)}) \\ &= P(A^t_{12}, \ldots, A^t_{1n}, \ldots, A^t_{n1}, \ldots, A^t_{n(n-1)}) = P(\vec{\mathcal{A}}^{t\intercal}) \ ,\end{aligned}$$

one can combine $P(\vec{X}^{0\intercal})$ with $P(\vec{\mathcal{A}}^{1\intercal})$ to $P(\vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal})$ s.t. the probability distribution of the first eliminated state variable X^1_{E1} can be combined as $P(X^1_{E1}|*, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(\vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) = P(X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal})$, and one obtains

$$\begin{aligned}P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) &= P(X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \\ &\quad \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1|\vec{X}^{1\intercal}\backslash\vec{X}^1, A^{1\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i\intercal}\backslash\vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ .\end{aligned}$$

Consecutively, there $\exists X^1_{E2} : \forall \{i\backslash E1\}\, A^1_{i(E2)} = \mathit{false}$ in every regular instantiation s.t.

$$\begin{aligned}P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) &= P(X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(X^1_{E2}|*, X^1_{E1}, *, \vec{A}^{1\intercal}_{E2}, X^0_{E2}) \\ &\quad \cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1|\vec{X}^{1\intercal}\backslash\vec{X}^1, A^{1\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i\intercal}\backslash\vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ ,\end{aligned}$$

for which one reverses the conditional independency again and obtain

$$\begin{aligned}P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) &= P(X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \cdot P(X^1_{E2}|*, X^1_{E1}, *, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \\ &\quad \cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1|\vec{X}^{1\intercal}\backslash\vec{X}^1, A^{1\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i\intercal}\backslash\vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ ,\end{aligned}$$

which, according to Bayes' chain rule, can be written as

$$\begin{aligned}P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) &= P(X^1_{E2}, X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \\ &\quad \cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1|\vec{X}^{1\intercal}\backslash\vec{X}^1, A^{1\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i\intercal}\backslash\vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ .\end{aligned}$$

Consecutively repeating this process for every remaining X_{Ei} where the i^{th} elimination variable is maximally dependent on the previous $(i-1)$ eliminated variables, one, henceforth, approaches the elimination of X^1_{En}, which is dependent on up to every other $\vec{X}^1$, which are all eliminated variables up to now, i.e.,

$$\begin{aligned}P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) &= P(X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1\intercal}, \vec{X}^{0\intercal}) \\ &\quad \cdot P(X^1_{En}|X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{A}^{1\intercal}_{En}, X^0_{En}) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i\intercal}\backslash\vec{X}^i, A^{i\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ ,\end{aligned}$$

for which one reverses the conditional independency again and combines the joint probability finally to

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = P(X^1_{En}, X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{i^\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ .$$

Indeed, one already obtains a topological ordering $>$ for the first two timeslices of $X^1_{En} > X^1_{E(n-1)} > \ldots > X^1_{E1} > \vec{\mathcal{A}}^{1^\intercal} > \vec{X}^{0^\intercal}$.
Following this procedure for the remaining $t > 1$, one finally obtains

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = P(X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \cdot P(X^t_{En} | X^t_{E(n-1)}, \ldots, X^1_{E1}, A^{1^\intercal}_{En}, X^{t-1}_{En}) \ .$$

With a final reverse conditional independence assumption,

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = P(X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \cdot P(X^t_{En} | X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \ ,$$

one obtains a complete topological ordering and a full joint probability over all random variables of

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) \quad = \quad P(X^t_{En}, X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \ . \quad (11)$$

One, thus, obtains a complete topological ordering that syntactically defines an equivalent Bayesian network (unrolled dynamic Bayesian network) $(B_0, B'_{\rightarrow})$ with the same random variables $\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t}$, i.e., defines the same full joint probability distribution $P(X^t_{En}, X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal})$. Therefore, the product of all locally defined CPDs, in fact, is the semantics of a dense intra-timeslice ADBN, despite being based on a cyclic graph and does not require any global normalization factors. □

Appendix B. Proof of Filtering Equation

We prove both Theorems 2 and 3 by showing that an algorithm exists which solves each problem, and which has the acclaimed computational complexity. Theorem 2 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{OffFP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$ in time-complexity $\mathcal{O}(t \cdot |\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\operatorname{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\operatorname{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and with space-complexity $\mathcal{O}(t \cdot |\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ for storing $P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$ for every timeslice t. We prove Theorem 2 by reducing it to multiple online filtering problems.

Theorem 3 is proven by showing that an algorithm algorithm exists that finds an exact solution with time-complexity $\mathcal{O}(|\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\operatorname{dom}(\zeta_+^{t-1})|^{|\vec{\zeta}^{t-1}|} \cdot |\operatorname{dom}(\beta_+^{t-1})|^{|\vec{\beta}^{t-1}|})$ and space-complexity $\mathcal{O}(|\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$. We prove Theorem 3 by deriving the general filtering equation from the joint probability distribution equation, which, seen as an algorithm, has the acclaimed time- and space-complexity for both filtering problems.

Proof of Theorems 2 and 3 (filtering). With a normalization factor α, filtering is generally defined from the joint probability as

$$P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \alpha \cdot \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) \ , \tag{12}$$

where all values of variables $\vec{X}^t, \vec{\mathcal{A}}^t$ are defined by the query and variables $\vec{X}^{0:t-1}, \vec{\mathcal{A}}^{1:t-1}$ are defined by either observations in the sets $\vec{z}^{0:t-1}, \vec{b}^{1:t-1}$ or through summation over unobserved variables in $\vec{\zeta}^{0:t-1}, \vec{\beta}^{1:t-1}$.

Using the recursive definition of the joint probability in Equation 5, one obtains

$$\begin{aligned}
&P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \\
&= \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} P(\vec{X}^{0:t-1\intercal}, \vec{\mathcal{A}}^{1:t-1\intercal}) \qquad \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \backslash X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&= \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} \sum_{\vec{\zeta}^{0:t-2}} \sum_{\vec{\beta}^{1:t-2}} P(\vec{X}^{0:t-1\intercal}, \vec{\mathcal{A}}^{1:t-1\intercal}) \\
&\qquad \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \backslash X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&= \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \backslash X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&\qquad \cdot \sum_{\vec{\zeta}^{0:t-2}} \sum_{\vec{\beta}^{1:t-2}} P(\vec{X}^{0:t-1\intercal}, \vec{\mathcal{A}}^{1:t-1\intercal}) \\
&= \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \backslash X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&\qquad \cdot P(\vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal} | \vec{z}^{0:t-1\intercal}, \vec{b}^{1:t-1\intercal}) \ .
\end{aligned}$$

One obtains the general filtering equation as

$$\begin{aligned}
P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} &P(\vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal} | \vec{z}^{0:t-1\intercal}, \vec{b}^{1:t-1\intercal}) \\
&\cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \backslash X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \ .
\end{aligned} \tag{13}$$

The general filtering equation reuses a previously stored filtering results from $t-1$. Evaluating $P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ by Equation 13 for all instantiations of $\vec{X}^t, \vec{\mathcal{A}}^t$, using a stored solution to all $P(\vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal} | \vec{z}^{0:t-1\intercal}, \vec{b}^{1:t-1\intercal})$, is an algorithm that gives an exact solution to the online filtering problem $\texttt{OnlFP}(B_0, B_\rightarrow, \vec{z}^{t-1:t}, \vec{b}^{t-1:t}, t)$. For a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{t-1:t}, \vec{b}^{t-1:t}$ the algorithm has constant time- and space-complexity $\mathcal{O}(1)$.

Iteratively evaluating Equation 13 for all instantiations of $\vec{X}^i, \vec{\mathcal{A}}^i, \forall i : 0 \leq i \leq t$, is an algorithm that gives an exact solution to the offline filtering problem $\texttt{OffFP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$. For a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{1:t}, \vec{b}^{1:t}$ the algorithm has linear time- and space-complexity $\mathcal{O}(t)$.

To obtain a complete distribution, both algorithms require exponentially many evaluations in the dimension of all possible instantiations of random variables per timeslice. Every incremental evaluation is exponential in the number and domain size of unobserved variables from the previous timeslice. □

Appendix C. Proof of Smoothing Equation

We prove both Theorems 4 and 5 by showing that an algorithm exists which solves each problem, and which has the acclaimed computational complexity. Theorem 4 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathcal{b}}^{1:t}, t)$ in time-complexity $\mathcal{O}(t^2 \cdot |\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\operatorname{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\operatorname{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and with $\mathcal{O}(t \cdot |\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ space-complexity for storing all smoothing distributions. Theorem 5 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{k-1:t}, \vec{\mathcal{b}}^{k-1:t}, \Delta, t)$ with time-complexity $\mathcal{O}(\Delta \cdot |\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\operatorname{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\operatorname{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and $\mathcal{O}(|\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ space-complexity for storing a smoothing distribution of timeslice k

Proof of Theorems 5 and 4 (smoothing). The general smoothing equation is obtained by straight marginalization from the joint probability as

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal}) = \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:t\intercal}, \vec{\mathcal{A}}^{1:t\intercal}) \ .$$

Using Equation 5 one obtains

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal}) = \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:t-1\intercal}, \vec{\mathcal{A}}^{1:t-1\intercal}) \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \backslash X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \quad . \quad (14)$$

We define an intermediate joint probability as

$$P(\vec{X}^{k:t\intercal}, \vec{\mathcal{A}}^{k:t\intercal}) = \prod_{X_i^k \in \vec{X}^k} P(X_i^k | \vec{X}^{k\intercal} \backslash X_i^k, \vec{A}_i^{k\intercal}, X_i^{k-1}) \cdot \prod_{A_{ij}^k \in \vec{\mathcal{A}}^k} P(A_{ij}^k) \cdot P(\vec{X}^{k+1:t\intercal}, \vec{\mathcal{A}}^{k+1:t\intercal}) \ ,$$

which is a term $T^{k-1:t}$ depending on variables from time $k-1$ until t. One can therefore split the recursive definition of the joint probability and obtain

$$\begin{aligned} &P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{\mathcal{b}}^{1:t\intercal}) \\ &= \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:k\intercal}, \vec{\mathcal{A}}^{1:k\intercal}) \cdot P(\vec{X}^{k+1:t\intercal}, \vec{\mathcal{A}}^{k+1:t\intercal}) \\ &= \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} P(\vec{X}^{0:k\intercal}, \vec{\mathcal{A}}^{1:k\intercal}) \quad \cdot \quad \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t\intercal}, \vec{\mathcal{A}}^{k+1:t\intercal}) \end{aligned}$$

$$= \alpha \cdot \left(\sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} P(\vec{X}^{0:k\intercal}, \vec{\mathcal{A}}^{1:k\intercal}) \right) \cdot \left(\sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t\intercal}, \vec{\mathcal{A}}^{k+1:t\intercal}) \right) . \quad (15)$$

One finds a previous (stored) filtering problem in the first sum-product, known as a "forward message," and a new latter sum-product commonly known in smoothing equations as a "backward-message." Using an adequate recursive definition for the latter term, one obtains the so-called forward-backward algorithm. The commonly known "sensor model" is, due to in-timeslice dependencies, included in the forward, as well as backward message.

The latter sum-product term from $k+1$ to t corresponds to $P(\vec{z}^{k+1:t\intercal}, \vec{b}^{k+1:t\intercal} | \vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal})$ from a probabilistically point of view, but essentially is a just a term $T^{k:t}$ over variables from time k to t. The splitting into two disjoint products is possible, because all variables from time k are query variables, i.e., are constant in a query. Without determining a *full* joint probability, a backward message is not derivable in multiply connected BNs. In fact, the probabilistic view also derives from Bayes' theorem and a first-order Markov assumption, but we rely on straight calculus here, as intra-timeslice dependencies pose various pitfalls here. The term is obtained as

$$\begin{aligned}
&P(\vec{z}^{k+1:t\intercal}, \vec{b}^{k+1:t\intercal} | \vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}) \\
&\quad = \alpha \cdot \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t\intercal}, \vec{\mathcal{A}}^{k+1:t\intercal}) \\
&\quad = \alpha \cdot \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1\intercal} \setminus X_i^{k+1}, \vec{A}_i^{k+1\intercal}, X_i^k) \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \\
&\qquad\qquad \cdot P(\vec{X}^{k+2:t\intercal}, \vec{\mathcal{A}}^{k+2:t\intercal}) \\
&\quad = \alpha \cdot \sum_{\vec{\zeta}^{k+1}} \sum_{\vec{\beta}^{k+1}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1\intercal} \setminus X_i^{k+1}, \vec{A}_i^{k+1\intercal}, X_i^k) \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \\
&\qquad\qquad \cdot \sum_{\vec{\zeta}^{k+2:t}} \sum_{\vec{\beta}^{k+2:t}} P(\vec{X}^{k+2:t\intercal}, \vec{\mathcal{A}}^{k+2:t\intercal}) , \quad (16)
\end{aligned}$$

in which one finds the backward message as the latter term. Using a recursive definition, one obtains the backward message as

$$\begin{aligned}
&P(\vec{z}^{k+1:t\intercal}, \vec{b}^{k+1:t\intercal} | \vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}) \\
&\qquad = \alpha \cdot \sum_{\vec{\zeta}^{k+1}} \sum_{\vec{\beta}^{k+1}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1\intercal} \setminus X_i^{k+1}, \vec{A}_i^{k+1\intercal}, X_i^k) \\
&\qquad\qquad \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \cdot P(\vec{z}^{k+2:t\intercal}, \vec{b}^{k+2:t\intercal} | \vec{X}^{k+1\intercal}, \vec{\mathcal{A}}^{k+1\intercal}) \quad . \quad (17)
\end{aligned}$$

Finally, by combining Equation 15 and 17, one obtains the general smoothing equation as

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{b}^{1:k\intercal}) \cdot P(\vec{z}^{k+1:t\intercal}, \vec{b}^{k+1:t\intercal} | \vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal}) . \quad (18)$$

With the recursive definition of the backward message, Eq. 17, this yields

$$P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \alpha \cdot P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{b}^{1:k\intercal}) \cdot \sum_{\vec{\zeta}^{k+1}} \sum_{\vec{\beta}^{k+1}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1\intercal} \setminus X_i^{k+1}, \vec{A}_i^{k+1\intercal}, X_i^k) \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \cdot P(\vec{z}^{k+2:t\intercal}, \vec{b}^{k+2:t\intercal} | \vec{X}^{k+1\intercal}, \vec{\mathcal{A}}^{k+1\intercal}) \ . \quad (19)$$

Evaluating the general smoothing Equation 18 for all instantiations of $\vec{X}^k, \vec{\mathcal{A}}^k$ decrementally for descending $k = t-1 \dots 0$ is an algorithm that gives an exact solution to the complete smoothing problem $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$. During the decremental evaluation the backward message is stored: an intermediate result $P(\vec{z}^{k+1:t\intercal}, \vec{b}^{k+1:t\intercal} | \vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal})$ (the backward message) obtained during an evaluation of $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ at time k is needed in an upcoming (but temporally backward) evaluation of $P(\vec{X}^{k-1\intercal}, \vec{\mathcal{A}}^{k-1\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ at time $k-1$. Thus, obtaining the last term of Equation 19 is constant in t for every evaluation. The first term $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{b}^{1:k\intercal})$ of the general smoothing equation poses a filtering problem, for which a solution is found in $\mathcal{O}(1)$ in a storage from a solution to an offline filtering problem $\texttt{OffFP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$, which is found in $\mathcal{O}(t)$ by the previously derived algorithm in Proof B. If memory is scarce, $P(\vec{X}^{k\intercal}, \vec{\mathcal{A}}^{k\intercal} | \vec{z}^{0:k\intercal}, \vec{b}^{1:k\intercal})$ is obtained in $\mathcal{O}(k)$ for every k by solving all $\texttt{OnlFP}(B_0, B_\rightarrow, \vec{z}^{i-1:i}, \vec{b}^{i-1:i}, i), 0 \leq i \leq k$. Thus, for a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{1:t}, \vec{b}^{1:t}$ the algorithm has linear time-complexity $\mathcal{O}(t)$ and linear space-complexity $\mathcal{O}(2 \cdot t)$ or quadratic time-complexity $\mathcal{O}(t^2)$, but halved space-complexity $\mathcal{O}(t)$. For every evaluation at time k, the backward message "moves one down" from $k+1$ to k and the forward message "ripples up" from 0 to k (or is obtained from a storage). For special DPGMs, a forward message can also be "moved one down" by using an inverse matrix calculation. With this picture in mind, the forward-backward-algorithm is sometimes seen as a "dance" between messages.

Evaluating the general smoothing Equation 18 for all instantiations of $\vec{X}^k, \vec{\mathcal{A}}^k$ for $k = t - \Delta$ is an algorithm that gives an exact solution to the fixed lag smoothing problem $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{t-\Delta-1:t}, \vec{b}^{t-\Delta-1:t}, \Delta, t)$. For a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{t-\Delta-1:t}, \vec{b}^{t-\Delta-1:t}$, the algorithm obtains the first term as a solution to an online filtering problem in $\mathcal{O}(1)$ and the latter term in $\mathcal{O}(\Delta)$ using the recursive definition.

To obtain a complete distribution, both algorithms require exponentially many evaluations in the dimension of all possible instantiations of random variables per timeslice. Every incremental evaluation is exponential in the number and domain size of unobserved variables from the previous and consecutive timeslice. □

Appendix D. Proof of Approximation Procedure

Correctness of Algorithm 2 (SIS). We prove that for $n_S \to \infty$, Theorem 6 provides an exact answer to a filtering query $P(\vec{x}^t, \vec{a}^t | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal})$ for the SIS case of Algorithm 2 in a dense intra-timeslice ADBN.

Notation 9 (Nostalgic sample)**.** *Let $\mathfrak{S} = \left(\vec{x}^{0:t-1}, \vec{a}^{1:t-1}, \vec{x}^t, \vec{a}^t, w\right)$ denote a sample of all random variables from time 0 to time t. This means that sample $\mathfrak{S}$ carries its complete history $\vec{x}^{0:t-1}, \vec{a}^{1:t-1}$ that ends in an end-sample $S = \left(\vec{x}^t, \vec{a}^t, w\right)$. Respectively, let $\mathfrak{S}^t_{\vec{x}^{0:t}, \vec{a}^{1:t}}$ denote a sample with evolution sequence $\vec{x}^{0:t}, \vec{a}^{1:t}$ and let $\vec{\mathfrak{S}}^t_{\vec{x}^{0:t}, \vec{a}^{1:t}}$ denote the set of all samples $\mathfrak{S}^t_{\vec{x}^{0:t}, \vec{a}^{1:t}}$ obtained at time t.*

Using Notation 9, Eq. 9 is written as

$$P(\vec{x}^{t\intercal}, \vec{a}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \approx \frac{\sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} \sum_{\mathfrak{S} \in \vec{\mathfrak{S}}^t_{\vec{x}^{0:t}, \vec{a}^{1:t}}} w_{\mathfrak{S}}}{\sum_{S \in \vec{S}^t} w_S} \ ,$$

i.e., to sum over all samples $S_{\vec{x}^t, \vec{a}^t}$, one now sums over all possible histories leading to such samples. As some instantiations are fixed by evidence in $\vec{z}^{0:t}, \vec{b}^{1:t}$, the summation over all unobserved variables leads to all possible evolution sequences. Following Algorithm 1 it is evident that the weight $w_{\mathfrak{S}}$ of a nostalgic sample $\mathfrak{S}$ is uniquely determined by its evolution instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$, i.e., all samples in $\vec{\mathfrak{S}}^t_{\vec{x}^{0:t}, \vec{a}^{1:t}}$ carry the weight

$$w_{\vec{x}^{0:t}, \vec{a}^{1:t}} = \prod_{k=0}^{t} \prod_{A^k_{ij} \in \vec{b}^k} P(A^k_{ij}) \cdot \prod_{X^k_i \in \vec{z}^k} P(X^k_i | \vec{X}^{k\intercal} \backslash X^k_i, A^{k\intercal}_i, X^{k-1}_i) \ . \tag{20}$$

Note that a weight w_S of a sample $S_{\vec{S}^t_{\vec{x}, \vec{a}}}$ is not uniquely determined by its instantiation $\vec{x}^t, \vec{a}^t$. Let $N^t_{\vec{x}^{0:t}, \vec{a}^{1:t}} = |\vec{\mathfrak{S}}^t_{\vec{x}^{0:t}, \vec{a}^{1:t}}|$ be the number of nostalgic samples at time t with evolution sequence $\vec{x}^{0:t}, \vec{a}^{1:t}$. Then

$$P(\vec{x}^{t\intercal}, \vec{a}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) \approx \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} N^t_{\vec{x}^{0:t}, \vec{a}^{1:t}} \cdot w_{\vec{x}^{0:t}, \vec{a}^{1:t}} \ , \tag{21}$$

with a normalization factor $\alpha = \sum_{S \in \vec{S}^t} w_S$. For $n_S \to \infty$, the number of a sample S, $N^t_{\mathfrak{S}}$, equals their existence probability $P(\mathfrak{S})$. Algorithm 1 generates a nostalgic sample $\mathfrak{S} = \left(\vec{x}^{0:t}, \vec{a}^{1:t}, w\right)$ with probability

$$P(\mathfrak{S}) = \prod_{k=0}^{t} \prod_{A^k_{ij} \in \vec{\beta}^k} P(A^k_{ij}) \cdot \prod_{X^k_i \in \vec{\zeta}^k} P(X^k_i | \vec{X}^{k\intercal} \backslash X^k_i, A^{k\intercal}_i, X^{k-1}_i) \ . \tag{22}$$

For brevity, let Eq. 20 and Eq. 22 ignore the special case of $t = 0$. By combining Equations 20–22, one obtains

$$\begin{aligned} P(\vec{x}^{t\intercal}, \vec{a}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} \prod_{k=0}^{t} \prod_{A^k_{ij} \in \vec{\beta}^k} P(A^k_{ij}) \\ \cdot \prod_{X^k_i \in \vec{\zeta}^k} P(X^k_i | \vec{X}^{k\intercal} \backslash X^k_i, A^{k\intercal}_i, X^{k-1}_i) \cdot \prod_{k=0}^{t} \prod_{A^k_{ij} \in \vec{b}^k} P(A^k_{ij}) \cdot \prod_{X^k_i \in \vec{z}^k} P(X^k_i | \vec{X}^{k\intercal} \backslash X^k_i, A^{k\intercal}_i, X^{k-1}_i) \ , \end{aligned}$$

as $\vec{Z}^{0:t} \cap \vec{\zeta}^{0:t} = \vec{X}^{0:t}$ and $\vec{\mathcal{B}}^{1:t} \cap \vec{\beta}^{1:t} = \vec{\mathcal{A}}^{1:t}$ this is

$$P(\vec{x}^{t\intercal}, \vec{a}^{t\intercal} | \vec{z}^{0:t\intercal}, \vec{b}^{1:t\intercal}) = \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} \prod_{k=0}^{t} \prod_{A^k_{ij} \in \vec{\mathcal{A}}^k} P(A^k_{ij}) \cdot \prod_{X^k_i \in \vec{X}^{\,k}} P(X^k_i | \vec{X}^{k\intercal} \backslash X^k_i, A^{k\intercal}_i, X^{k-1}_i) \tag{23}$$

which corresponds to the exact solution for the filtering problem in Eq. 12 with the complete unrolled joint probability from Eq. 4. □

References

Acid, S., & de Campos, L. M. (2003). Searching for Bayesian Network Structures in the Space of Restricted Acyclic Partially Directed Graphs. *Journal of Artificial Intelligence Research*, *18*, 445–490.

Arulampalam, M. S., Maskell, S., Gordon, N. J., & Clapp, T. (2002). A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, *50*(2), 174–188.

Bilmes, J. A. (2000). Dynamic Bayesian Multinets. In *UAI 2000: 16th Conference on Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000*, pp. 38–45.

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). Context-Specific Independence in Bayesian Networks. In *UAI 1996: 12th Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996*, pp. 115–123.

Cooper, G. F. (1990). The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence*, *42*(2-3), 393–405.

de Raedt, L., Kimmig, A., & Toivonen, H. (2007). ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI 2007: 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pp. 2462–2467.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., & Zhang, W. (2014). Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *KDD 2014: 20th ACM International Conference on Knowledge Discovery and Data Mining, New York, New York, USA, August 24-27, 2014*, pp. 601–610.

Doucet, A., de Freitas, N., Murphy, K. P., & Russell, S. J. (2000). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *UAI 2000: 16th Conference on Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000*, pp. 176–183.

Doucet, A., & Johansen, A. M. (2009). A Tutorial on Particle Filtering and Smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, *12*, 656–704.

Drton, M. (2009). Discrete Chain Graph Models. *Bernoulli*, *15*(3), 736–753.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., & Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, *31*(3), 59–79.

Fierens, D., den Broeck, G. V., Renkens, J., Shterionov, D. S., Gutmann, B., Thon, I., Janssens, G., & de Raedt, L. (2015). Inference and Learning in Probabilistic Logic Programs Using Weighted Boolean Formulas. *Theory and Practice of Logic Programming*, *15*(3), 358–401.

Geiger, D., & Heckerman, D. (1996). Knowledge Representation and Inference in Similarity Networks and Bayesian Multinets. *Artificial Intelligence*, *82*(1-2), 45–74.

Ghahramani, Z. (2001). An Introduction to Hidden Markov Models and Bayesian Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, *15*(1), 9–42.

Gibbs, A. L., & Su, F. E. (2002). On Choosing and Bounding Probability Metrics. *International Statistical Review*, *70*(3), 419–435.

Glesner, S., & Koller, D. (1995). Constructing Flexible Dynamic Belief Networks from First-Order Probabilistic Knowledge Bases. In *ECSQARU 1995: Symbolic and Quantitative Approaches to Reasoning and Uncertainty, European Conference, Fribourg, Switzerland, July 3-5, 1995*, pp. 217–226.

Haddawy, P., Helwig, J., Ngo, L., & Krieger, R. (1995). Clinical Simulation using Context-Sensitive Temporal Probability Models. In *Symposium on Computer Applications in Medical Care*, Vol. 1, pp. 203–207.

Heckerman, D., & Breese, J. S. (1996). Causal Independence for Probability Assessment and Inference Using Bayesian Networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, *26*(6), 826–831.

Henrion, M. (1988). Practical Issues in Constructing a Bayes Belief Network. *International Journal of Approximate Reasoning*, *2*(3), 337.

Hol, J. D., Schön, T. B., & Gustafsson, F. (2006). On Resampling Algorithms for Particle Filters. In *NSSPW 2006: IEEE Nonlinear Statistical Signal Processing Workshop, Cambridge, UK, September 13-15, 2006*, pp. 79–82. IEEE.

Jaeger, M. (2001). Complex Probabilistic Modeling with Recursive Relational Bayesian Networks. *Annals of Mathematics and Artificial Intelligence*, *32*(1-4), 179–220.

Levesque, H. J., Davis, E., & Morgenstern, L. (2012). The Winograd Schema Challenge. In *KR 2012: 13th International Conference on Principles of Knowledge Representation and Reasoning, Rome, Italy, June 10-14, 2012*.

Massey, B. (2008). Fast Perfect Weighted Resampling. In *ICASSP 2008: IEEE International Conference on Acoustics, Speech, and Signal Processing, Caesars Palace, Las Vegas, Nevada, USA, March 30 - April 4, 2008*, pp. 3457–3460.

Milch, B., Marthi, B., Sontag, D., Russell, S. J., Ong, D. L., & Kolobov, A. (2005). Approximate Inference for Infinite Contingent Bayesian Networks. In *AISTATS 2005: 10th International Workshop on Artificial Intelligence and Statistics, Bridgetown, Barbados, January 6-8, 2005*.

Motzek, A. (2016). *Indirect Causes, Dependencies and Causality in Bayesian Networks*. Ph.D. thesis, University of Luebeck, Germany. available: http://adbn.motzek.org/adbn-com-rev.pdf.

Motzek, A., & Möller, R. (2015a). Exploiting Innocuousness in Bayesian Networks. In *AI 2015: 28th Australasian Joint Conference on Artificial Intelligence, Canberra, ACT, Australia, November 30 - December 4, 2015*, pp. 411–423.

Motzek, A., & Möller, R. (2015b). Indirect Causes in Dynamic Bayesian Networks Revisited. In *IJCAI 2015: 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, 2015*, pp. 703–709.

Motzek, A., & Möller, R. (2017). Context- and Bias-Free Probabilistic Mission Impact Assessment. *Computers & Security*, *65*, 166–186.

Motzek, A., Möller, R., Lange, M., & Dubus, S. (2015). Probabilistic Mission Impact Assessment based on Widespread Local Events. In *NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, NATO IST-128 Workshop, Istanbul, Turkey, June 15-17, 2015*, pp. 16–22.

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA.

Ngo, L., & Haddawy, P. (1997). Answering Queries from Context-Sensitive Probabilistic Knowledge Bases. *Theoretical Computer Science*, *171*(1-2), 147–177.

Ngo, L., Haddawy, P., & Helwig, J. (1995). A Theoretical Framework for Context-Sensitive Temporal Probability Model Construction with Application to Plan Projection. In *UAI 1995: 11th Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pp. 419–426.

Nodelman, U., Shelton, C. R., & Koller, D. (2002). Continuous Time Bayesian Networks. In *UAI 2002: 18th Conference on Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002*, pp. 378–387.

Pearl, J., & Russell, S. (2003). Bayesian Networks. In Arbib, M. A. (Ed.), *Handbook of Brain Theory and Neural Networks*, pp. 157–160. MIT Press.

Pfeffer, A., & Tai, T. (2005). Asynchronous Dynamic Bayesian Networks. In *UAI 2005: 21st Conference on Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pp. 467–476.

Poole, D., & Zhang, N. L. (2003). Exploiting Contextual Independence In Probabilistic Inference. *Journal Of Artificial Intelligence Research*, *18*, 263–313.

Robinson, J. W., & Hartemink, A. J. (2008). Non-Stationary Dynamic Bayesian Networks. In *NIPS 2008: 22nd Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 1369–1376.

Robinson, J. W., & Hartemink, A. J. (2010). Learning Non-Stationary Dynamic Bayesian Networks. *Journal of Machine Learning Research*, *11*, 3647–3680.

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education.

Sanghai, S., Domingos, P., & Weld, D. (2005). Relational Dynamic Bayesian Networks. *Journal of Artificial Intelligence Research*, *24*, 759–797.

Sloane, N. J. A. (2015). The On-Line Encyclopedia of Integer Sequences. http://oeis.org/. OEIS Foundation Inc. Sequences A003024 & A001831.

Song, L., Kolar, M., & Xing, E. P. (2009). Time-Varying Dynamic Bayesian Networks. In *NIPS 2009: 23rd Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 7-10, 2009*, pp. 1732–1740.